Integrator/CM946E-S
Integrator/CM966E-S
User Guide

ARM

Copyright © 2000 ARM Limited. All rights reserved.
ARM DUI 0138A

Integrator/CM946E-S
Integrator/CM966E-S
User Guide

Copyright © ARM Limited 2000. All rights reserved.

Release Information

Description	Issue	Change
26 September 2000	A	New document

Proprietary Notice

ARM, the ARM Powered logo, Thumb, and StrongARM are registered trademarks of ARM Limited.

The ARM logo, AMBA, Angel, ARMulator, EmbeddedICE, ModelGen, Multi-ICE, PrimeCell, ARM7TDMI, ARM7TDMI-S, ARM9TDMI, ARM9E-S, ARM946E-S, ARM966E-S, ETM7, ETM9, TDMI, and STRONG are trademarks of ARM Limited.

All other products or services mentioned herein may be trademarks of their respective owners.

Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder.

The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. However, all warranties implied or expressed, including but not limited to implied warranties of merchantability, or fitness for purpose, are excluded.

This document is intended only to assist the reader in the use of the product. ARM Limited shall not be liable for any loss or damage arising from the use of any information in this document, or any error or omission in such information, or any incorrect use of the product.

Conformance Notices

This section contains *ElectroMagnetic Conformity* (EMC) notices.

Federal Communications Commission Notice

NOTE: This equipment has been tested and found to comply with the limits for a class A digital device, pursuant to part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

CE Declaration of Conformity

This equipment has been tested according to ISE/IEC Guide 22 and EN 45014. It conforms to the following product EMC specifications:

The product herewith complies with the requirements of EMC Directive 89/336/EEC as amended.

Confidentiality Status

This document is Open Access. This document has no restriction on distribution.

Product Status

The information in this documents is Final (information on a developed product).

Web Address

`http://www.arm.com`

Copyright © 2000 ARM Limited. All rights reserved.

Copyright © 2000 ARM Limited. All rights reserved.

Contents
Integrator/CM946E-S
Integrator/CM966E-S
User Guide

Preface
About this document viii
Further reading x
Feedback xi

Chapter 1 Introduction
1.1 About the ARM Integrator/CM9x6E-S core module 1-2
1.2 ARM Integrator/CM9x6E-S overview 1-4
1.3 Links and indicators 1-8
1.4 Test points and connectors 1-10
1.5 Precautions 1-12

Chapter 2 Getting Started
2.1 Setting up a standalone ARM Integrator/CM9x6E-S 2-2
2.2 Attaching the ARM Integrator/CM9x6E-S to a motherboard 2-5

Chapter 3 Hardware Description
3.1 ARM microprocessor test chip 3-2

Copyright © 2000 ARM Limited. All rights reserved.

3.2 SSRAM controller 3-5
3.3 Core module FPGA 3-6
3.4 SDRAM controller 3-8
3.5 Reset controller 3-10
3.6 System bus bridge 3-12
3.7 Module ID selection 3-18
3.8 Clock generators 3-20
3.9 Multi-ICE support 3-24
3.10 Embedded Trace support 3-33

Chapter 4 Programmer's Reference
4.1 About the ARM9x6E-S memory map 4-2
4.2 Core module memory map configuration 4-4
4.3 SSRAM alias 4-8
4.4 SDRAM mapping 4-9
4.5 Processor configuration 4-11
4.6 Core module control registers 4-12
4.7 Core module flag registers 4-25
4.8 Core module interrupt registers 4-26
4.9 SDRAM SPD memory 4-30

Appendix A Signal Descriptions
A.1 HDRA A-2
A.2 HDRB A-4
A.3 Trace connector pinout A-9
A.4 Logic analyzer connectors A-10

Appendix B Specifications
B.1 Electrical specification B-2
B.2 Timing specification B-3
B.3 Mechanical details B-6

Index

 Copyright © 2000 ARM Limited. All rights reserved.

Preface

This preface introduces the ARM Integrator/CM9x6E-S core module and its reference documentation. It contains the following sections:

- *About this document* on page viii
- *Further reading* on page x
- *Feedback* on page xi.

Copyright © 2000 ARM Limited. All rights reserved.

About this document

This document describes how to set up and use the ARM Integrator/CM9x6E-S core module.

Intended audience

This document has been written for experienced hardware and software developers to aid the development of ARM-based products using the ARM Integrator/CM966E-S or CM946E-S as part of a development system.

Organization

This document is organized into the following chapters:

Chapter 1 *Introduction*

Read this chapter for an introduction to the core module.

Chapter 2 *Getting Started*

Read this chapter for a description of how to set up and start using the core module.

Chapter 3 *Hardware Description*

Read this chapter for a description of the hardware architecture of the core module. This includes clocks, resets, and debug.

Chapter 4 *Programmer's Reference*

Read this chapter for a description of the core module memory map and registers.

Appendix A *Signal Descriptions*

Refer to this appendix for connector pinouts.

Appendix B *Specifications*

Refer to this appendix for electrical, timing, and mechanical specifications.

Copyright © 2000 ARM Limited. All rights reserved.

Typographical conventions

The following typographical conventions are used in this document:

bold	Highlights ARM processor signal names within text, and interface elements such as menu names. This style is also used for emphasis in descriptive lists where appropriate.
italic	Highlights special terminology, cross-references and citations.
`typewriter`	Denotes text that can be entered at the keyboard, such as commands, file names and program names, and source code.
`typewriter`	Denotes a permitted abbreviation for a command or option. The underlined text can be entered instead of the full command or option name.
`typewriter italic`	Denotes arguments to commands or functions where the argument is to be replaced by a specific value.
`typewriter bold`	Denotes language keywords when used outside example code.

Copyright © 2000 ARM Limited. All rights reserved.

Further reading

This section lists related publications by ARM Limited and other companies that might provide additional information.

ARM publications

The following publications provide information about related ARM products and toolkits:

- *ARM966E-S Technical Reference Manual* (ARM DDI 0164)
- *ARM946E-S Technical Reference Manual* (ARM DDI 0155)
- *ARM9 Embedded Trace Module (ETM9) Technical Reference Manual* (ARM DUI 0157)
- *ARM Integrator/SP User Guide* (ARM DUI 0099)
- *ARM Integrator/AP User Guide* (ARM DUI 0098)
- *ARM Multi-ICE User Guide* (ARM DUI 0048)
- *AMBA Specification* (ARM IHI 0011)
- *ARM Architectural Reference Manual* (ARM DDI 0100)
- *ARM Firmware Suite Reference Guide* (ARM DUI 0102)
- *ARM Software Development Toolkit User Guide* (ARM DUI 0040)
- *ARM Software Development Toolkit Reference Guide* (ARM DUI 0041)
- *ADW Trace Debug Tools User Guide* (ARM DUI 0118)
- *ADS Tools Guide* (ARM DUI 0067)
- *ADS Debuggers Guide* (ARM DUI 0066)
- *ADS Debug Target Guide* (ARM DUI 0058)
- *ADS Developer Guide* (ARM DUI 0056)
- *ADS CodeWarrior IDE Guid*e (ARM DUI 0065).

Other publications

The following publication provides information about the clock controller chip used on the Integrator modules:

- *MicroClock OSCaR User Configurable Clock Data Sheet* (MDS525), MicroClock Division of ICS, San Jose, CA.

The following publications provide information and guidelines for developing products for Microsoft Windows CE:

- *Standard Development Board for Microsoft® Windows® CE*, 1998, Microsoft Corporation.

Copyright © 2000 ARM Limited. All rights reserved.

Feedback

ARM Limited welcomes feedback both on the ARM Integrator/CM9x6E-S core module and on the documentation.

Feedback on this document

If you have any comments about this document, please send email to `errata@arm.com` giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

Feedback on the ARM Integrator/CM966E-S and CM946E-S

If you have any comments or suggestions about these products, please contact your supplier giving:

- the product name
- an explanation of your comments.

Copyright © 2000 ARM Limited. All rights reserved.

 Copyright © 2000 ARM Limited. All rights reserved.

Chapter 1
Introduction

This chapter introduces the ARM Integrator/CM9x6E-S core module. It contains the following sections:

- *About the ARM Integrator/CM9x6E-S core module* on page 1-2
- *ARM Integrator/CM9x6E-S overview* on page 1-4
- *Links and indicators* on page 1-8
- *Test points and connectors* on page 1-10
- *Precautions* on page 1-12.

Copyright © 2000 ARM Limited. All rights reserved.

1.1 About the ARM Integrator/CM9x6E-S core module

The Integrator/CM9x6E-S core module is a compact development platform that enables you to develop products based on the ARM966E-S or ARM946E-S processors. It provides you with a flexible development environment which can be configured in the following ways:

- as a standalone development system using Multi-ICE for program download
- mounted onto an ARM Integrator motherboard
- integrated into a third-party development or ASIC prototyping system.

Through-board connectors allow up to four core modules to be stacked on one motherboard.

Figure 1-1 on page 1-3 shows the layout of the ARM Integrator/CM9x6E-S.

Copyright © 2000 ARM Limited. All rights reserved.

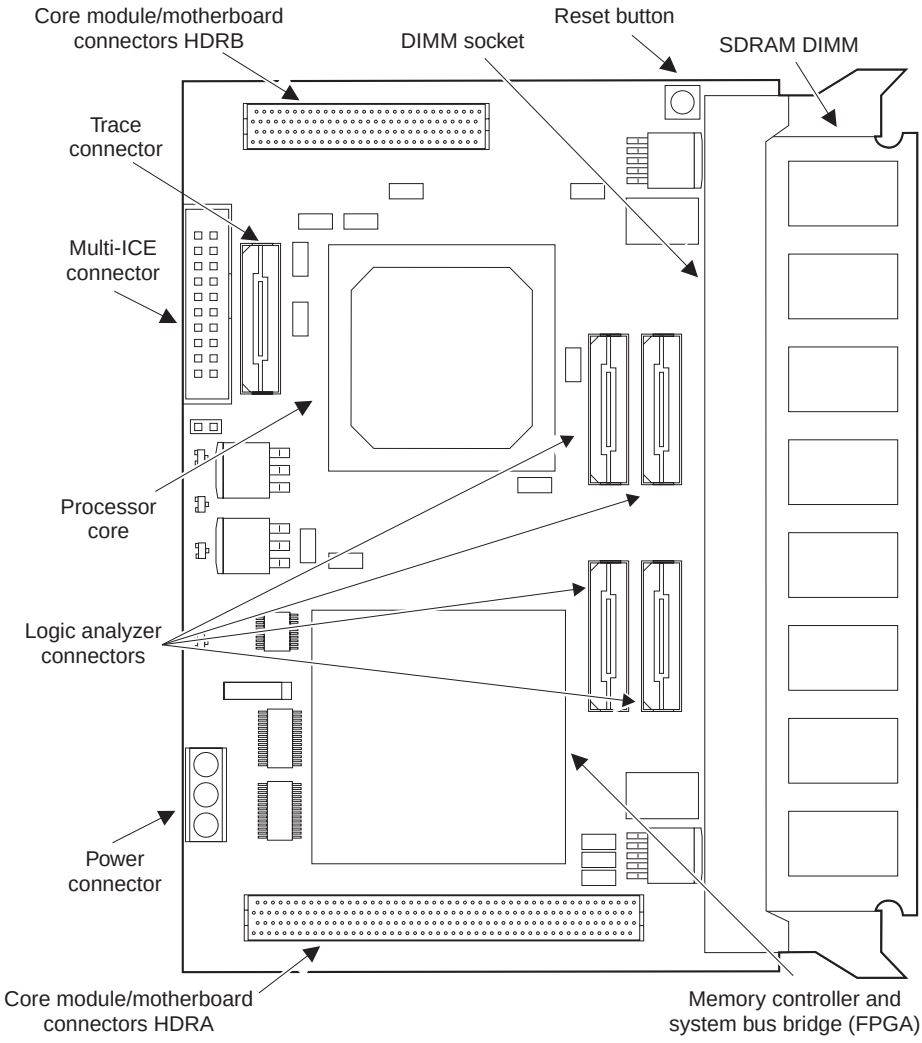


Figure 1-1 Integrator/ CM9x6E-S layout

Copyright © 2000 ARM Limited. All rights reserved.

1.2 ARM Integrator/CM9x6E-S overview

The major components on the core module are as follows:

- ARM966E-S or ARM946E-S microprocessor core
- volatile memory comprising:
  - up to 256MB of SDRAM (optional) plugged into the DIMM socket
  - 1MB SSRAM.
- core module FPGA which implements:
  - SDRAM controller
  - system bus bridge
  - reset controller
  - interrupt controller
  - status, configuration, and interrupt registers.
- SSRAM controller PLD
- clock generator
- Integrator system bus connectors
- Multi-ICE debug connector
- logic analyzer connectors for local memory bus
- Trace port.

Copyright © 2000 ARM Limited. All rights reserved.

1.2.1 System architecture

Figure 1-2 illustrates the architecture of the core module.

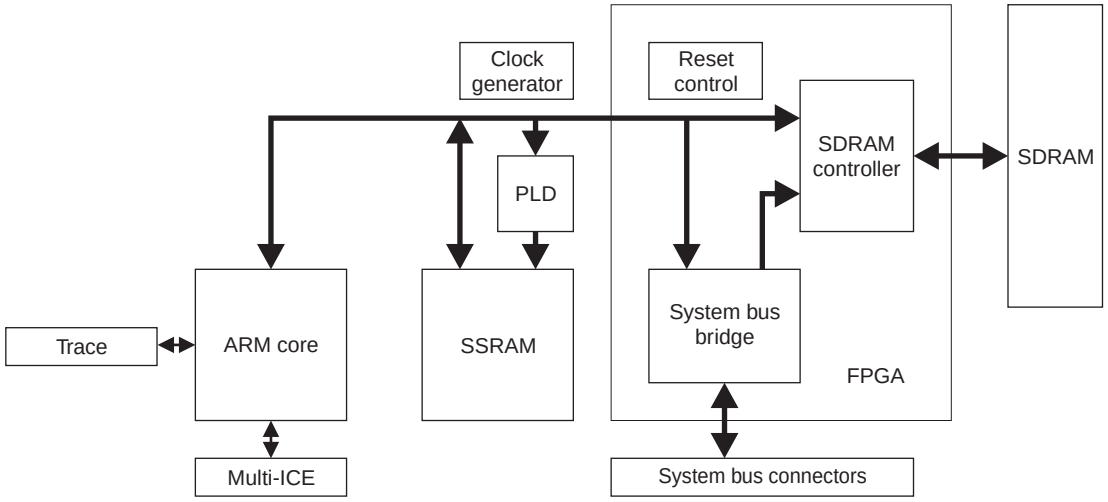


Figure 1-2 ARM Integrator/CM9x6E-S block diagram

1.2.2 ARM processor test chip

The Integrator/CM946E-S is fitted with an ARM946E-S and the Integrator/CM966E-S is fitted with an ARM966E-S. For a brief description of these processors, see *ARM microprocessor test chip* on page 3-2.

You can configure both of these processor types using a number of input signals. In a normal product, these signals are permanently tied HIGH or LOW. However, the Integrator allows you to reprogram these inputs in order to experiment with different processor configurations (see *Test chip configuration control* on page 3-2).

1.2.3 Core module FPGA

The FPGA provides system control functions for the core module, enabling it to operate as a standalone development system or attached to a motherboard. These functions are outlined in this section and described in detail in Chapter 3 *Hardware Description*.

Copyright © 2000 ARM Limited. All rights reserved.

SDRAM controller

The SDRAM controller is implemented within the FPGA. This provides support for *Dual In-line Memory Modules* (DIMMs) with a capacity of between 16 and 256MB. See SDRAM controller on page 3-8.

Reset controller

The reset controller initializes the core. The core module can be reset from five sources:

- reset button
- motherboard
- other core modules
- Multi-ICE
- software.

For information about the reset controller (see *Reset controller* on page 3-10).

System bus bridge

The system bus bridge provides an AMBA AHB interface between the memory bus on the core module and the system bus on a motherboard. It allows the processor to access resources on the motherboard and on other modules. It also allows other masters to access the core module SDRAM (see *System bus bridge* on page 3-12).

Status and configuration space

The status and configuration space contains status and configuration registers for the core module. These provide the following information and control:

- processor status and configuration
- the position of the core module in a multi-module stack
- SDRAM size, address configuration, and CAS latency setup
- core module clock speed and configuration
- interrupt control for the processor debug communications channel.

The status and control registers can only be accessed by the local processor. For more information about the status and control registers (see Chapter 4 *Programmer's Reference*).

 Copyright © 2000 ARM Limited. All rights reserved.

1.2.4 Volatile memory

The volatile memory system includes an SSRAM device, and a plug-in SDRAM memory module (referred to as *local* SDRAM when it is on the same core module as the processor). These areas of memory are on the local memory bus to ensure high performance. The core module uses separate memory and system buses to avoid memory access performance being degraded by bus loading.

The SDRAM controller is implemented within the core module controller FPGA and a separate SSRAM controller is implemented with a *Programmable Logic Device* (PLD).

The SSRAM can only be accessed by the local processor.

Note

The ARM946E-S and ARM966E-S both feature tightly coupled SRAM inside the test chip (see *About the ARM9x6E-S memory map* on page 4-2).

1.2.5 Clock generation

The core module generates three clock signals:

REFCLK A fixed frequency 24MHz signal that can be used by the FPGA to generate real-time delays.

ARM_PLLCLKIN A programmable frequency reference clock input to the PLL in the ARM test chip.

AUXCLK A programmable frequency clock reserved for future use.

The programmable clocks are supplied by two clock generator chips. Their frequencies are set by programming the oscillator control registers within the FPGA. A fixed-frequency reference clock is supplied to the two clock generators and to the FPGA (see *Clock generators* on page 3-20).

The memory bus and system bus are asynchronous. This allows each bus to be run at the speed of its slowest device without compromising the performance of other buses in the system.

1.2.6 Multi-ICE connector

The Multi-ICE connector enables JTAG hardware debugging equipment, such as Multi-ICE, to be connected to the core module. You can drive and sense the system-reset line (**nSRST**), and drive JTAG reset (**nTRST**) to the core from the Multi-ICE connector (see *Multi-ICE support* on page 3-24).

Copyright © 2000 ARM Limited. All rights reserved.

1.3 Links and indicators

The core module provides one link and four surface-mounted LEDs. These are illustrated in Figure 1-3.

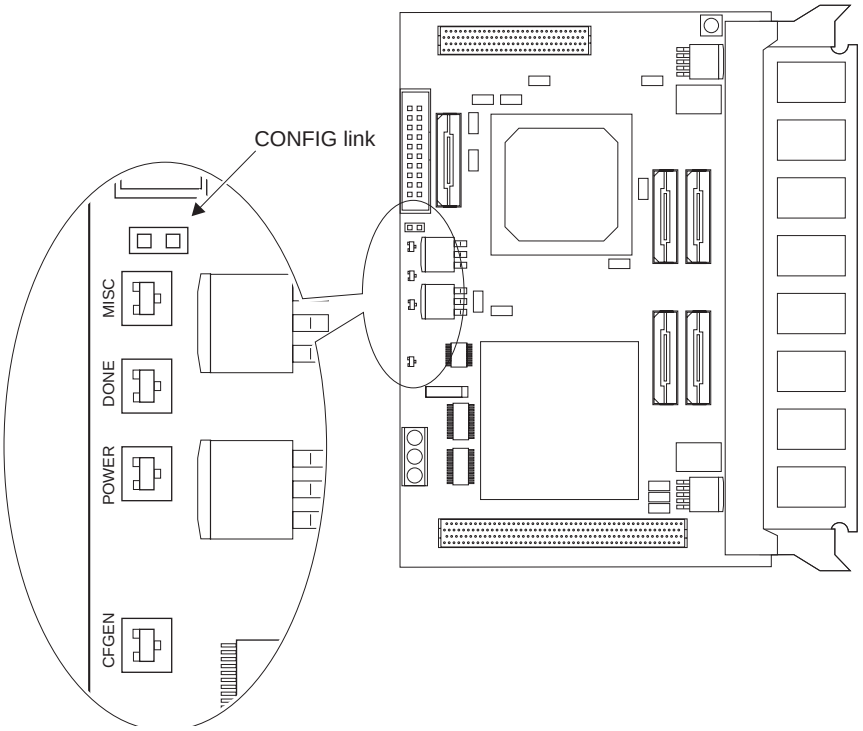


Figure 1-3 Links and indicators

1.3.1 CONFIG link

The core module has only one link, known as the CONFIG link (shown in Figure 1-3). This is left open during normal operation. Only fit this link to download new FPGA and PLD configurations (see *Multi-ICE support* on page 3-24).

Copyright © 2000 ARM Limited. All rights reserved.

1.3.2 LED indicators

The functions of the four surface-mounted LEDs are summarized in Table 1-1.

Table 1-1 LED functional summary

Name	Color	Function
MISC	Green	This LED is controlled by the LED bit in the CM_CTRL register (see *Core module control register* on page 4-15).
DONE	Green	This LED is lit when the FPGA has successfully loaded its configuration information following power-on.
POWER	Green	This LED is let to indicate that a 3.3V supply is present.
CFGEN	Orange	This LED is lit to indicate that the CONFIG link is fitted. Fitting this link to the core module at the top of the stack places all modules in the stack in CONFIG mode (see Debugging modes on page 3-23).

Copyright © 2000 ARM Limited. All rights reserved.

1.4 Test points and connectors

Figure 1-4 shows the test points and ground points on the core module.

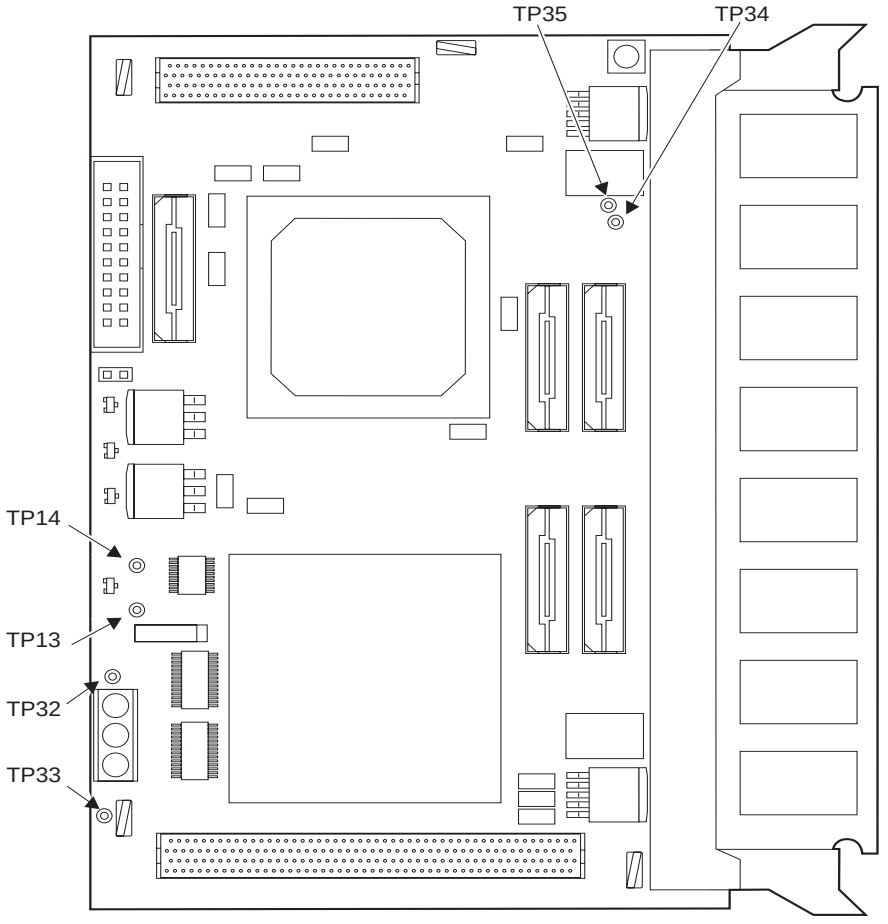


Figure 1-4 Test points

Copyright © 2000 ARM Limited. All rights reserved.

The functions of the test points are summarized in Table 1-2.

Table 1-2 Test point functions

Test point	Name	Function
TP13	PLLCKIN	Input clock to ARM9x6E-S
TP14	REF CLK	Reference clock (24MHz)
TP32	AUX CLK	Auxiliary clock
TP33	PLL LOCK	Phase locked (if supported by the test chip)
TP34	SDRAM HCLK	Buffered **HCLK** to SDRAM DIMM
TP35	HCLK	**HCLK** output from test chip

Copyright © 2000 ARM Limited. All rights reserved.

1.5 Precautions

This section contains safety information and advice on how to avoid damage to the core module.

1.5.1 Ensuring safety

The core module is powered from 3.3V and 5V DC supplies.

Warning

To avoid a safety hazard, only *Safety Extra Low Voltage* (SELV) equipment should be connected to the JTAG interface.

1.5.2 Preventing damage

The core module is intended for use within a laboratory or engineering development environment. It is supplied without an enclosure which leaves the board sensitive to electrostatic discharges and allows electromagnetic emissions.

Caution

To avoid damage to the board you must observe the following precautions.

- Never subject the board to high electrostatic potentials.
- Always wear a grounding strap when handling the board.
- Only hold the board by the edges.
- Avoid touching the component pins or any other metallic element.

Caution

Do not use the board near equipment that could be:

- sensitive to electromagnetic emissions (such as medical equipment)
- a transmitter of electromagnetic emissions.

Copyright © 2000 ARM Limited. All rights reserved.

Chapter 2
Getting Started

This chapter describes how to set up and prepare the ARM Integrator/CM9x6E-S core module for use. It contains the following sections:

- *Setting up a standalone ARM Integrator/CM9x6E-S* on page 2-2
- *Attaching the ARM Integrator/CM9x6E-S to a motherboard* on page 2-5.

Copyright © 2000 ARM Limited. All rights reserved.

2.1 Setting up a standalone ARM Integrator/CM9x6E-S

To set up the core module as a standalone development system:

1. Optionally, fit an SDRAM DIMM.
2. Connect Multi-ICE.
3. Supply power.

2.1.1 Fitting an SDRAM DIMM

You can fit the following type of SDRAM module:

- PC66- or PC100-compliant 168pin DIMM
- unbuffered
- 3.3V
- 16MB, 32MB, 64MB, 128MB, or 256MB.

To install an SDRAM DIMM:

1. Ensure that the core module is powered down.
2. Open the SDRAM retaining latches outwards.
3. Press the SDRAM module into the edge connector until the retaining latches click into place.

Note

The DIMM edge connector has polarizing notches to ensure that it is correctly oriented in the socket.

2.1.2 Using the core module without SDRAM

You can operate the core module without SDRAM because it has 1MB of SSRAM permanently fitted. When using using ADW or AXD, you can adjust the `top_of_memory` internal variable from its default value to `0x100000`.

For further information about ARM debugger internal variables, refer to the *Software Development Toolkit Reference Guide* or *ADS Debuggers Guide*.

Copyright © 2000 ARM Limited. All rights reserved.

2.1.3 Connecting power

When using the core module as a standalone development system, you must connect a bench power supply with 3.3V and 5V outputs to the power connector, as illustrated in Figure 2-1. For information about power consumption by the core module, see *Electrical specification* on page B-2.

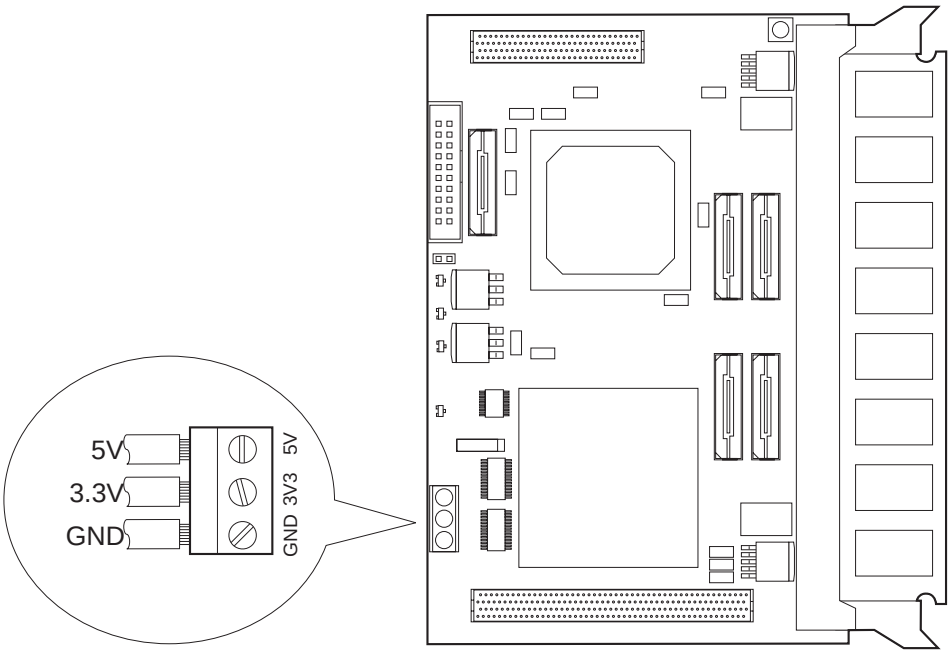


Figure 2-1 Power connector

Note

This power connection is not required when the core module is fitted to a motherboard.

Copyright © 2000 ARM Limited. All rights reserved.

2.1.4 Connecting Multi-ICE

When you are using the core module as a standalone system, Multi-ICE debugging equipment can be used to download programs. The Multi-ICE setup for a standalone core module is shown in Figure 2-2.

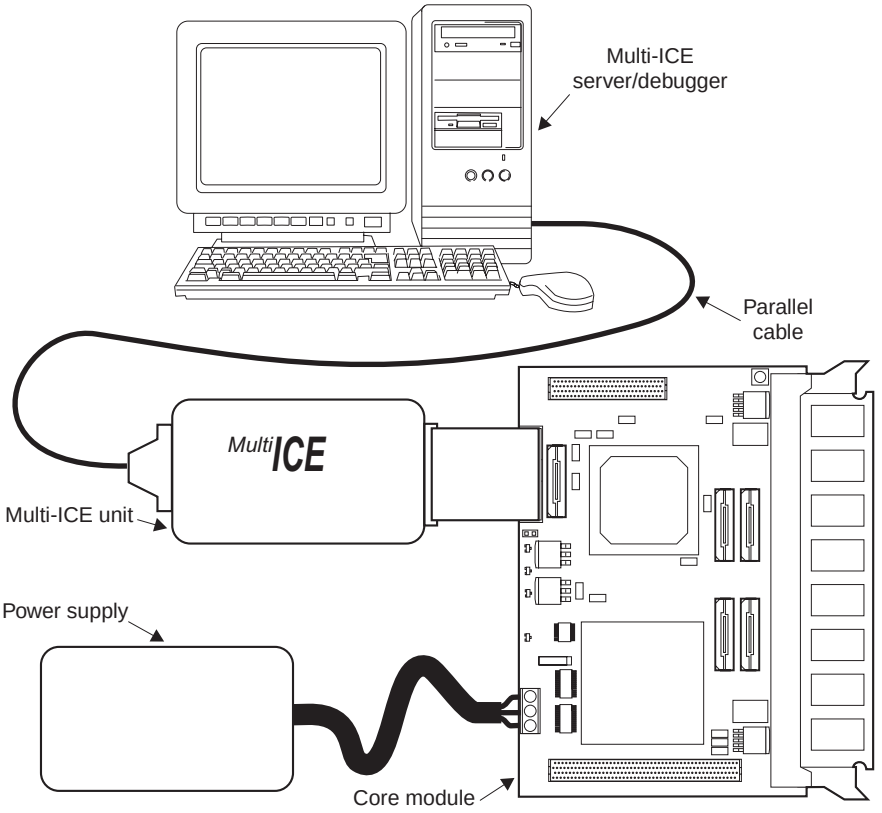


Figure 2-2 Multi-ICE connection to a core module

Caution

Because the core module does not provide nonvolatile memory, programs are lost when the power is removed.

You can also use Multi-ICE when a core module is attached to a motherboard. If more than one core module is attached, then the Multi-ICE unit must be connected to the module at the top of the stack. The Multi-ICE server and the debugger can be on one computer or on two networked computers.

Copyright © 2000 ARM Limited. All rights reserved.

2.2 Attaching the ARM Integrator/CM9x6E-S to a motherboard

Attach the core module onto a motherboard (for example, the ARM Integrator/SP) by engaging the connectors HDRA and HDRB on the bottom of the core module with the corresponding connectors on the top of the motherboard. The lower side of the core module has sockets and the upper side of the core module has plugs to allow core modules to be mounted on top of one another. A maximum of four core modules can be stacked on a motherboard.

Figure 2-3 illustrates an example development system with four core modules attached to an ARM Integrator/SP motherboard.

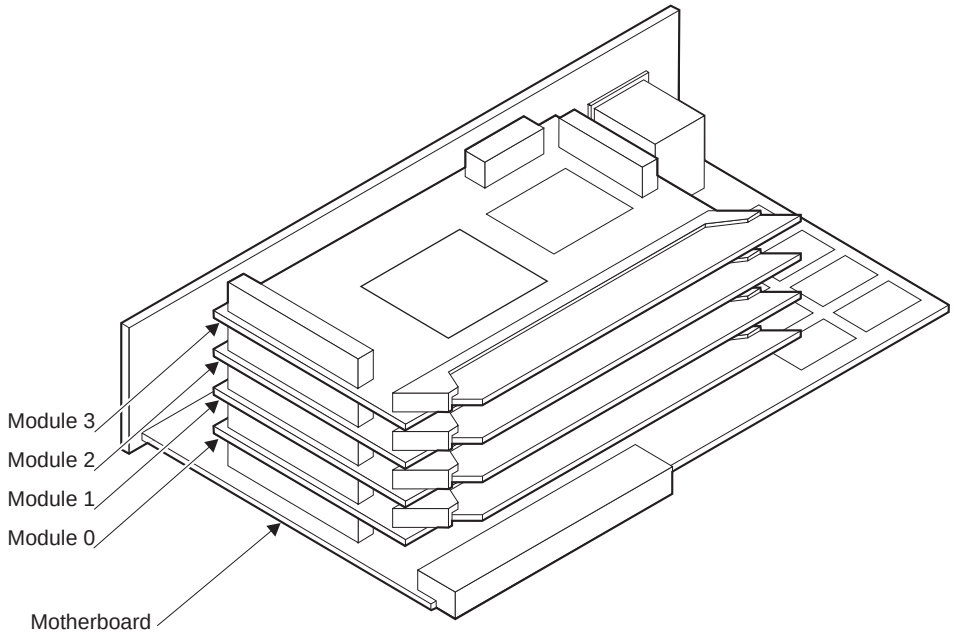


Figure 2-3 Assembled Integrator system

Note

To ensure reliable operation of the core module:

- Do not use the core module in the EXPA/EXPB stack position on the Integrator/AP.
- Configure the motherboard for AHB operation. The Integrator/CM946E-S and CM966E-S only support an AHB system bus interface.

Copyright © 2000 ARM Limited. All rights reserved.

2.2.1 Core module ID

The ID of the core module is configured automatically by the connectors (there are no links to set) and depends on its position in the stack:

- core module 0 is installed first
- core module 1 is installed next, and cannot be fitted without core module 0
- core module 2 is installed next, and cannot be fitted without core module 1
- core module 3 is installed next, and cannot be fitted without core module 2.

The ID of the core module also defines the address of SDRAM in the alias memory region of the system memory map. The mechanism that controls the ID and mapping of the core module is described in *Module ID selection* on page 3-18.

The position of a core module in the stack can be read from the CM_STAT register (see *Core module status register* on page 4-16).

2.2.2 Powering the assembled Integrator development system

Power the assembled Integrator development system by:

- connecting a bench power supply to the motherboard (not to the core module)
- installing the motherboard in a card cage or an ATX-type PC case, depending on type.

For further information, refer to the user guide for the motherboard you are using.

Copyright © 2000 ARM Limited. All rights reserved.

Chapter 3
Hardware Description

This chapter describes the on-board hardware. It contains the following sections:

- *ARM microprocessor test chip* on page 3-2
- *SSRAM controller* on page 3-5
- *Core module FPGA* on page 3-6
- *SDRAM controller* on page 3-8
- *Reset controller* on page 3-10
- *System bus bridge* on page 3-12
- *Module ID selection* on page 3-18
- *Clock generators* on page 3-20
- *Multi-ICE support* on page 3-24
- *Embedded Trace support* on page 3-33.

Copyright © 2000 ARM Limited. All rights reserved.

3.1 ARM microprocessor test chip

This section provides a brief overview of the ARM946E-S and ARM966E-S processor macrocells. It outlines their main features and differences, and describes the operational configuration options that they provide.

3.1.1 ARM9x6E-S overview

ARM946E-S and ARM966E-S processor macrocells are both based on the ARM9E-S RISC processor core. The ARM9E-S is a 32-bit RISC processor based on the ARM9TDMI core. It includes signal processing extensions to the ARM instruction set and a single-cycle 16 x 32 *Multiply-ACcumulate* (MAC) unit.

The ARM946E-S combines the ARM9E-S core with:

- instruction and data caches
- tightly coupled instruction and data SRAM
- write buffer
- memory protection unit.

The ARM966E-S combines the ARM9E-S core with:

- tightly coupled instruction and data SRAM
- write buffer.

Both macrocells are user code compatible with the ARM7TDMI, ARM9TDMI, StrongARM, and ARM1020T processors. They also support the instructions specific to the ARMv5TE architecture. The additional instructions improve DSP performance.

3.1.2 Test chip configuration control

The ARM9x6E-S processors are configurable and have a number of inputs signals that control the configuration of the ARM test chip. In a normal product, core configuration is static and these signals are tied HIGH or LOW as appropriate. However, because the Integrator is a development platform, it allows you to program the levels on these signals in the CM_INIT register for experimentation. The configuration signals driven by CM_INIT are described in Table 3-1 on page 3-3.

Note

The configuration inputs controlled by the CM_INIT register within the FPGA. CM_INIT bit-states are retained during normal resets. During *Power-On Reset* (POR), before the FPGA is configured, the bit-states are defaulted by pull-down resistors to the values shown in the POR default column.

 Copyright © 2000 ARM Limited. All rights reserved.

Table 3-1 Controllable processor configuration signals

Signal/Bit name	Function	POR default
INITRAM	Use this signal to enable (HIGH) or disable (LOW) the internal (tightly-coupled) SRAM.	LOW
VINITHI	Use this signal to initialize the vector base address to `0xFFFF0000` (HIGH) or `0x00000000` (LOW). The Integrator system memory map allocates the high vector address location as logic module expansion space which means that there may not be any physical memory at this location.	LOW
PLLBYPASS	Use this signal to control clock usage by processors that contain an internal PLL. When **PLLBYPASS** is HIGH, the processor uses the clock signal supplied on its **PLLCLKIN** signal pin. When **PLLBYPASS** is LOW, the processor uses the clock signal supplied by its internal PLL.	LOW

Changing the processor configuration

To change the configuration of the processor:

1. Program the appropriate values in the CM_INIT register (see *Core module initialization register* on page 4-22).
2. Reset the core module (but do not power-cycle) by pressing the reset button, or with a software reset, by writing a 1 to bit 3 of the CM_CTRL register (see *Core module control register* on page 4-15).

Restoring the default configuration

To restore the default processor configuration, power-cycle the core module.

Copyright © 2000 ARM Limited. All rights reserved.

3.1.3 Fixed value test chip configuration controls

The configuration inputs shown in Table 3-2 on page 3-4 cannot be changed.

Table 3-2 Fixed processor configuration signals

Signal	Function
EFIBYPASS	This signal enables (HIGH) or disables (LOW) a bypass of synchronizers for **nFIQ** and **nIRQ**. This signal is tied LOW.
RESBYPASS	This signal enables (HIGH) or disables (LOW) a bypass of the synchronizer for the **nRESET** signal. This signal is tied LOW.
MICEBYPASS	This signal enables (HIGH) or disables (LOW) enables a bypass of the synchronizer for the JTAG signal. This signal is tied LOW.

Copyright © 2000 ARM Limited. All rights reserved.

3.2 SSRAM controller

The SSRAM controller is implemented in a Xilinx 9572XL PLD. This enables the SSRAM to achieve single-cycle operation.

Copyright © 2000 ARM Limited. All rights reserved.

3.3 Core module FPGA

The core module FPGA contains five main functional blocks:

- *SDRAM controller* on page 3-8
- *Reset controller* on page 3-10
- *System bus bridge* on page 3-12
- *Core module control registers* on page 4-12
- Debug interrupt controller, see *Debug communications interrupts* on page 3-32.

The FPGA provides sufficient functionality for the core module to operate as a standalone development system, although with limited capabilities. System bus arbitration, system interrupt control, and input/output resources are provided by the system controller FPGA on the motherboard. See the user guide for your motherboard for further information.

Figure 3-1 illustrates the function of the core module FPGA and shows how it connects to the other devices in the system.

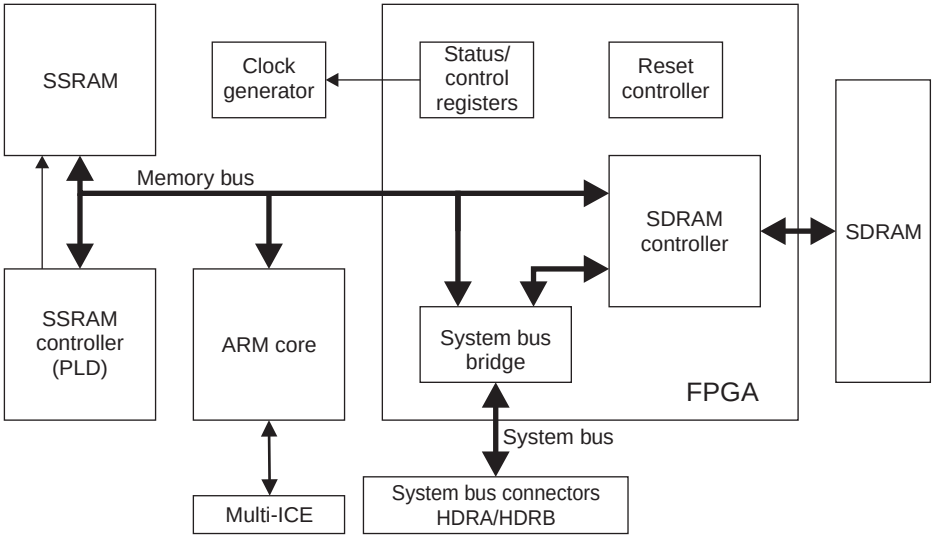


Figure 3-1 FPGA functional diagram

At power-up the FPGA loads its configuration data from a flash memory device. Parallel data from the flash is serialized by a *Programmable Logic Device* (PLD) into the configuration inputs of the FPGA. Figure 3-2 on page 3-7 shows the FPGA configuration mechanism.

Copyright © 2000 ARM Limited. All rights reserved.

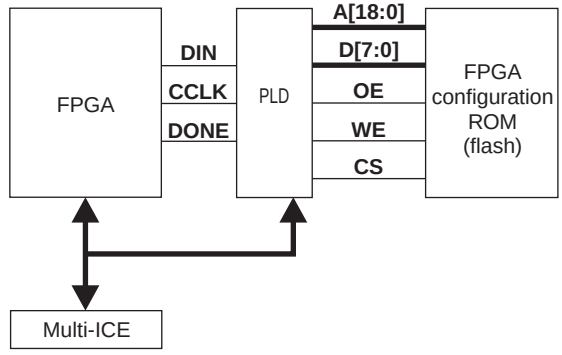


Figure 3-2 FPGA configuration

You can use Multi-ICE to reprogram the PLD, FPGA, and flash when the core module is placed in configuration mode (see *Multi-ICE support* on page 3-24).

Copyright © 2000 ARM Limited. All rights reserved.

3.4 SDRAM controller

The core module provides support for a single 16, 32, 64, 128, or 256MB SDRAM DIMM.

3.4.1 SDRAM operating mode

The operating mode of the SDRAM devices is controlled with the mode set register within each SDRAM. These registers are set immediately after power-up to specify:

- a burst size of four for both reads and writes
- *Column Address Strobe* (CAS) latency of 2 cycles.

Note

Before the SDRAM can be used, you must program the CAS latency and memory size parameters in the SDRAM control register (CM_SDRAM) at address `0x10000020`. The required parameters are read from the SPD memory and are listed in Table 4-18 on page 4-29. If you do not correctly set the SDRAM parameters, accesses could be slow or unreliable. See *SDRAM status and control register* on page 4-20.

3.4.2 Access arbitration

The SDRAM controller provides two ports to support reads and writes by the local processor core and by masters on the motherboard. The SDRAM controller uses an alternating priority scheme to ensure that the processor core and motherboard have equal access (see *System bus bridge* on page 3-12).

3.4.3 Serial presence detect

JEDEC-compliant SDRAM DIMMs incorporate a *Serial Presence Detect* (SPD) feature. This comprises a 2048-bit serial EEPROM located on the DIMM with the first 128 bytes programmed by the DIMM manufacturer to identify the following:

- module type
- memory organization
- timing parameters.

The EEPROM clock (SCL) operates at 93.75kHz (24MHz divided by 256). The transfer rate for read accesses to the EEPROM is 100kbit/s maximum. The data is read out serially 8 bits at a time, preceded by a start bit and followed by a stop bit. This makes reading the EEPROM a very slow process because it takes approximately 27ms to read all 256 bytes. However, during power-up the contents of the EEPROM are copied into a 64 x 32-bit area of memory (CM_SPD) within the SDRAM controller. The SPD flag

 Copyright © 2000 ARM Limited. All rights reserved.

is set in the SDRAM control register (CM_SDRAM) when the SPD data is available. This copy can be randomly accessed at `0x10000100` to `0x100001FC` (see *SDRAM SPD memory* on page 4-30).

Write accesses to the SPD EEPROM are not supported.

Copyright © 2000 ARM Limited. All rights reserved.

3.5 Reset controller

The core module FPGA incorporates a reset controller that enables the core module to be reset as a standalone unit or as part of an Integrator development system. The core module can be reset from five sources:

- reset button
- motherboard
- other core modules
- Multi-ICE
- software.

Figure 3-3 shows the architecture of the reset controller.

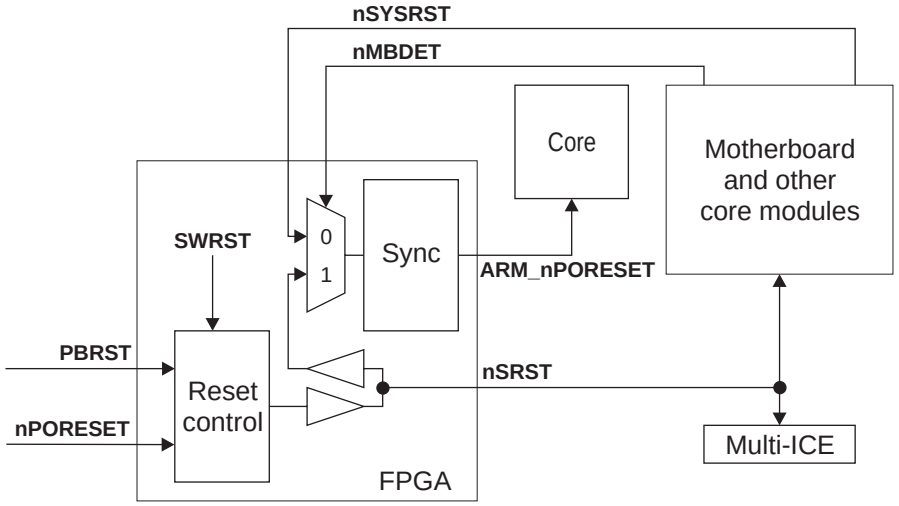


Figure 3-3 Core module reset controller

Copyright © 2000 ARM Limited. All rights reserved.

3.5.1 Reset control signals

Table 3-3 describes the external reset signals.

Table 3-3 Reset signal descriptions

Name	Description	Type	Function
ARM_nPORESET	Processor reset	Output	The **ARM_nPORESET** signal is used to reset the processor core. It is generated from **nSRST** LOW when the core module is used standalone, or **nSYSRST** LOW when the core module is attached to a motherboard. It is asserted as soon as the appropriate input becomes active. It is deasserted synchronously from the falling edge of the processor bus clock.
nMBDET	Motherboard detect	Input	The **nMBDET** signal is pulled LOW when the core module is attached to a motherboard and HIGH when the core module is used standalone. When **MBDET** is LOW, **nSYSRST** is used to generate the **ARM_nPORESET** signal. When **nMBDET** is HIGH, **nSRST** is used to generate the **ARM_nPORESET** signal.
PBRST	Push-button reset	Input	The **PBRST** signal is generated by pressing the reset button.
nSRST	System reset	Bidirectional	The **nSRST** open collector output signal is driven LOW by the core module FPGA when the signal **PBRST** or software reset (**SWRST**) is asserted. As an input, **nSRST** can be driven LOW by Multi-ICE. If there is no motherboard present, the **nSRST** signal is synchronized to the processor bus clock to generate the **ARM_nPORESET** signal.
nSYSRST	System reset	Input	The **nSYSRST** signal is generated by the system controller FPGA on the motherboard. It is used to generate the **ARM_nPORESET** signal when the core module is attached to a motherboard. It is selected by the motherboard detect signal (**nMBDET**).

3.5.2 Software resets

The core module FPGA provides a software reset which can be triggered by writing to the reset bit in the CM_CTRL register. This generates the internal reset signal **SWRST** which generates **nSRST** and resets the whole system (see *Core module control register* on page 4-15).

Copyright © 2000 ARM Limited. All rights reserved.

3.6 System bus bridge

The system bus bridge provides an asynchronous bus interface between the local memory bus and system bus connecting the motherboard and other modules. Accesses are supported by two 16-entry 74-bit FIFOs. Each of the 16 entries in the FIFOs contains:

- 32-bit data used for write transfers
- 32-bit address used for reads and writes
- 10-bit transaction control used for reads and writes.

3.6.1 Processor accesses to the system bus

The first FIFO supports accesses by the local processor to the motherboard and other modules on the system bus.

Processor writes

The data routing for processor writes to the system bus is illustrated in Figure 3-4.

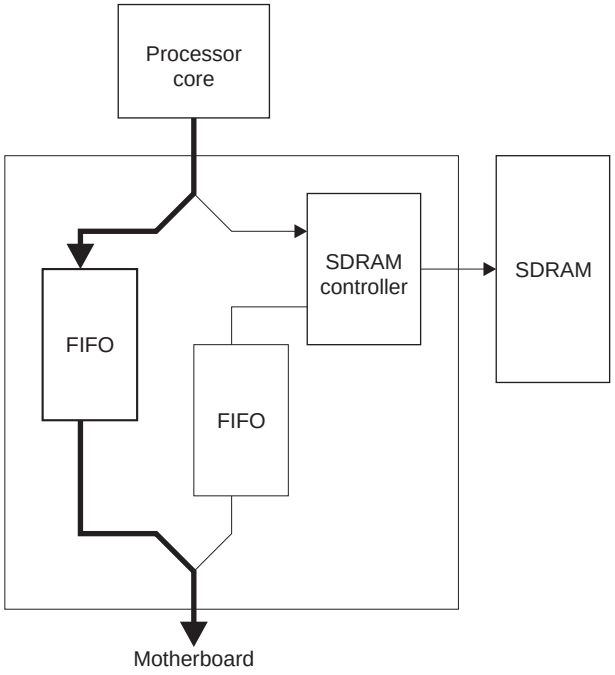


Figure 3-4 Processor writes to the system bus

Copyright © 2000 ARM Limited. All rights reserved.

Write transactions from the processor to the system bus normally complete on the local memory bus in a single cycle. The data, address, and control information associated with the transfer are posted into FIFO, and the transfer on the system bus occurs some time later when that bus is available. This means that system bus error responses to write transfers are not reported back to the processor as Data Aborts. If the FIFO is full, the processor receives a wait response until space becomes available.

Processor reads

The data routing for processor reads from the system bus is illustrated in Figure 3-5.

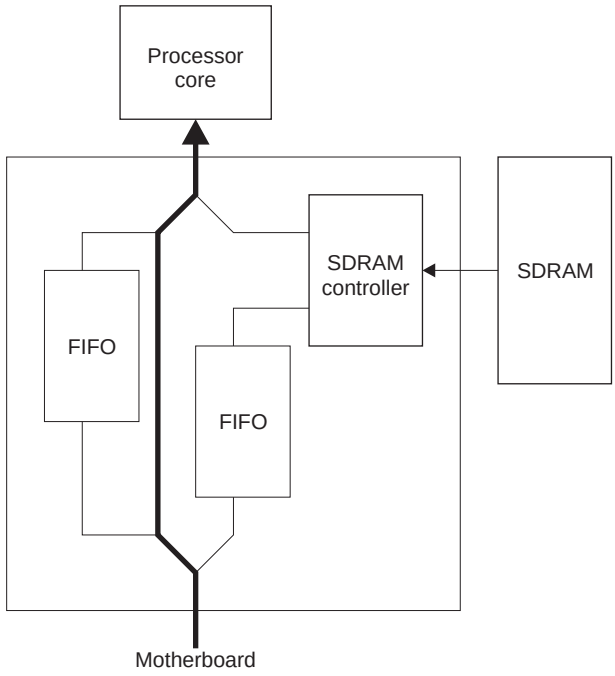


Figure 3-5 Processor reads from the system bus

For reads from the system bus, the address and control information pass through the FIFO, but the data from the system bus bypasses the FIFO.

The order of processor transactions is preserved on the system bus. Any previously posted writes are drained from the FIFO (completed on the system bus) before the read transfer is performed. The processor receives a wait response until the read transfer has completed on the system bus, after which it receives the data and any associated bus error response from the system bus. For information about SDRAM addresses (see *SDRAM mapping* on page 4-9).

Copyright © 2000 ARM Limited. All rights reserved.

You can use a system bus read to drain the FIFO of writes that would affect a subsequent operation. For example, to ensure that a write by an IRQ/FIQ handler clears an interrupt source, preventing the same interrupt being taken again by the core, the handler would follow a write to the interrupt source with a read from the system bus.

3.6.2 Motherboard accesses to SDRAM

The second FIFO supports read and write accesses by system bus masters on the motherboard and other core modules to the local core module memory.

System bus writes

The data routing for system bus writes to SDRAM is illustrated in Figure 3-6.

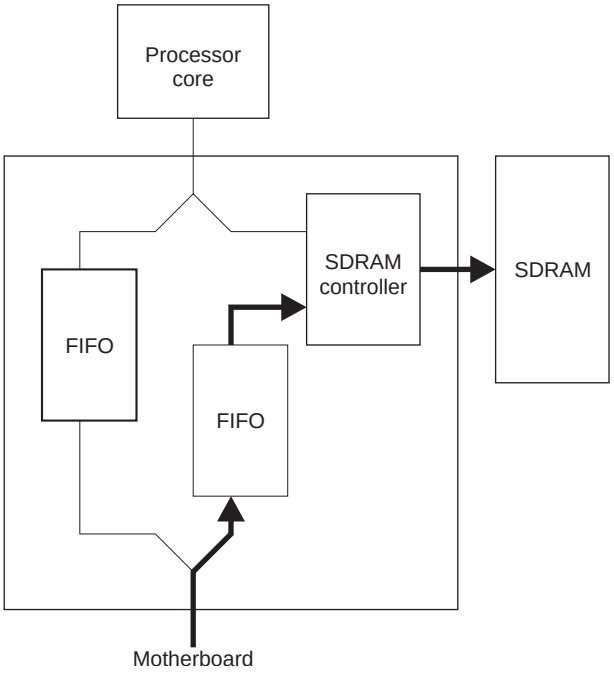


Figure 3-6 System bus writes to SDRAM

Write transactions from the system bus to the SDRAM normally complete in a single cycle on the system bus. The data, address, and control information associated with the transfer are posted into the FIFO, and the transfer into the SDRAM completes when the SDRAM is available. If the FIFO is full, then the system bus master receives a retract response indicating that the arbiter may grant the bus to another master and that this transaction must be retried later.

Copyright © 2000 ARM Limited. All rights reserved.

System bus reads

The data routing for system bus reads from SDRAM is illustrated in Figure 3-7.

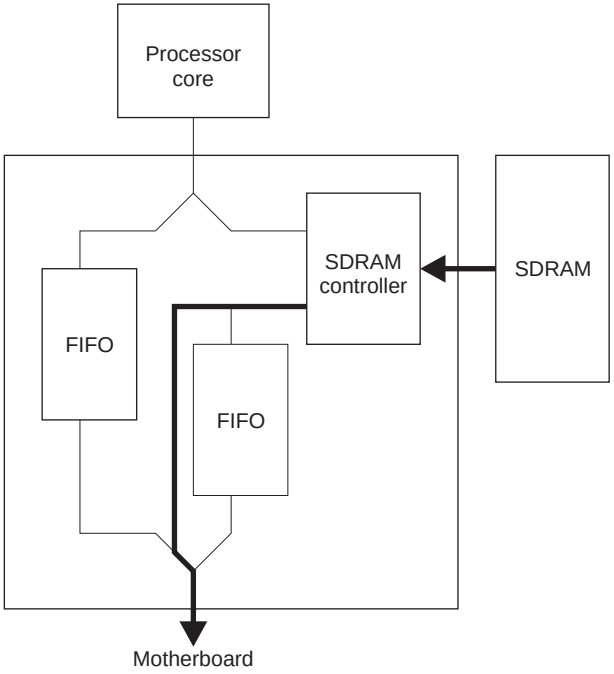


Figure 3-7 System bus reads from SDRAM

For system bus reads, the address and control information also pass through the FIFO, but the returned data from the SDRAM bypasses the FIFO.

The order of transactions on the system bus and the memory bus is preserved. Any previously posted write transactions are drained from the FIFO (that is, writes to SDRAM are completed) before the read transfer is performed.

3.6.3 Multiprocessor support

The two FIFOs operate independently and can be accessed at the same time. This makes it possible for a local processor to read local SDRAM over the system bus (through both FIFOs). This feature can be used to support multiprocessor systems that share data in SDRAM because the processors can access the same SDRAM locations.

Copyright © 2000 ARM Limited. All rights reserved.

3.6.4 System bus signal routing

The core module is mounted onto a motherboard using the connectors HDRA and HDRB. As well as carrying all signal connections between the boards, these provide mechanical mounting (see *Attaching the ARM Integrator/CM9x6E-S to a motherboard* on page 2-5).

HDRA

The signals on the HDRA connectors are tracked between the socket on the underside and the plug on the top so that pin 1 connects to pin 1, pin 2 to pin 2 and so on. That is, the signals are routed straight through.

HDRB

A number of signals on the HDRB connectors are rotated in groups of four between the connectors on the bottom and top of each module. This ensures that each processor (or other bus master device) on a module connects to the correct signals according to whether it is bus master 0, 1, 2, or 3. The ID for the bus master on a module is determined by the position of the module in the stack.

This signal rotation scheme is illustrated in Figure 3-8.

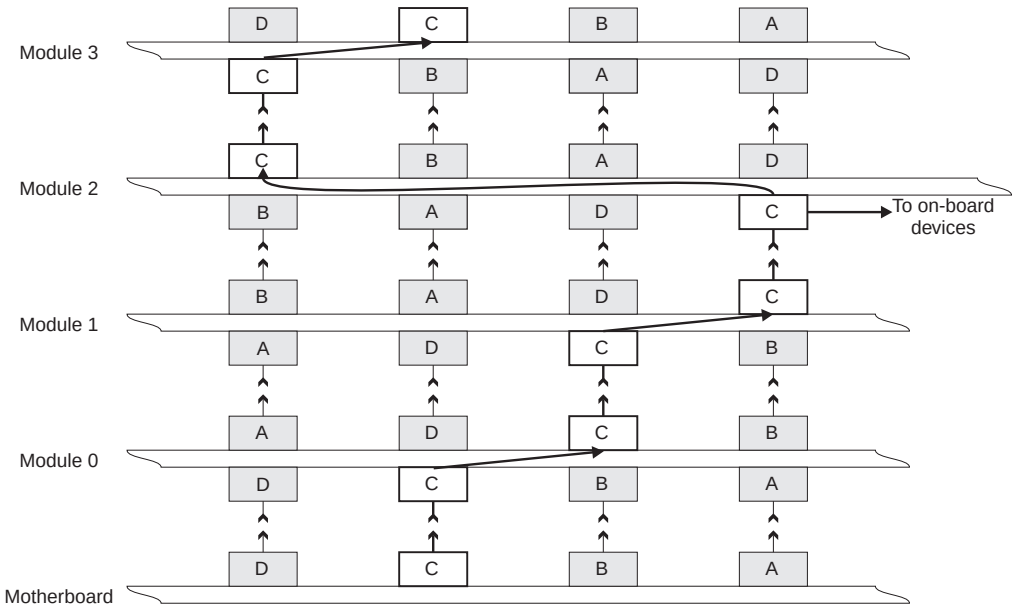


Figure 3-8 Signal rotation on HDRB

Copyright © 2000 ARM Limited. All rights reserved.

The example in Figure 3-8 on page 3-16 illustrates how a group of four signals (labelled A, B, C, and D) are routed through a group of four connector pins up through the stack. It highlights how signal C is rotated as it passes up through the stack and only utilized on module 2.

All four signals are rotated and utilized in a similar way, as follows:

- signal A on core module 0
- signal B on core module 1
- signal C used on core module 2
- signal D used on core module 3.

For details of the signals on the HDRB connectors, see*HDRB* on page A-4.

Note

The JTAG signals are discussed in *Multi-ICE support* on page 3-24.

Copyright © 2000 ARM Limited. All rights reserved.

3.7 Module ID selection

The position of a core module in the HDRA/HDRB stack is used to determine its ID, and from this its address in the alias memory region (see the user guide for your motherboard) and the interrupts that it responds to.

Note

The core module cannot be damaged by connecting it onto the EXPA/EXPB position on the Integrator/AP motherboard, but fitting it in this position prevents reliable operation.

3.7.1 Module address decoding

The Integrator system implements a distributed address decoding system. These means that each core or logic module must decode its own area of the memory map. The central decoder in the system controller FPGA (on the motherboard) responds with an error code for all areas of the address space that are not occupied by a module. This default response is disabled for a memory region occupied by a module that is fitted.

The signals **nPPRES[3:0]** (core module present) **nEPRES[3:0]** (logic module present) are used to signal the presence of modules to the central decoder. The signals **ID[3:0]** indicate to the module its position in the stack and the address range that its own decoder must respond to. These signals rotate as they pass up the stack, as described in *System bus signal routing* on page 3-16. Only one signal in each group is pulled LOW for each module.

The alias SDRAM address of a core module is determined in hardware, although a module can determine its own position by reading a decoded version of **ID[3:0]** from the CM_STAT register (see *Core module status register* on page 4-16). Table 3-4 shows alias addresses for a core module fitted to a the motherboard on the HDRA/HDRB stack.

Table 3-4 Core module address decode

ID[3:0]	Module ID	Address range	Size
1101	3 (top)	0xB0000000 to 0xBFFFFFFF	256MB
1011	2	0xA0000000 to 0xAFFFFFFF	256MB
0111	1	0x90000000 to 0x9FFFFFFF	256MB
1110	0 (bottom)	0x80000000 to 0x8FFFFFFF	256MB

Copyright © 2000 ARM Limited. All rights reserved.

3.7.2 Interrupts

The system controller FPGA on the motherboard incorporates an interrupt controller which routes the various interrupts from around the system onto the nFIQ and nIRQ pins of up to four processors. The interrupts that a core module receives are determined by the position of the core module within the stack, as shown in Table 3-5.

Table 3-5 Core module interrupts

Module ID	Interrupt	Fast interrupt
3 (top)	**nIRQ3**	**nFIQ3**
2	**nIRQ2**	**nFIQ2**
1	**nIRQ1**	**nFIQ1**
0 (bottom)	**nIRQ0**	**nFIQ0**

The interrupt signals are routed to the core module using pins on the HDRB connectors (see *HDRB* on page A-4).

The interrupts and fast interrupts are enabled and handled using the interrupt control registers on the motherboard (see the user guide for your motherboard).

Copyright © 2000 ARM Limited. All rights reserved.

3.8 Clock generators

The core module provides two programmable clock sources:

- the processor clock **ARM_PLLCLKIN**
- the auxiliary clock **AUXCLK**.

These two clocks are supplied by two MicroClock ICS525 clock generators, as shown in Figure 3-9. In addition, the core module produces the fixed-frequency clock **REFCLK** and a buffered version of the processor clock **FPGA_PLLCKIN**.

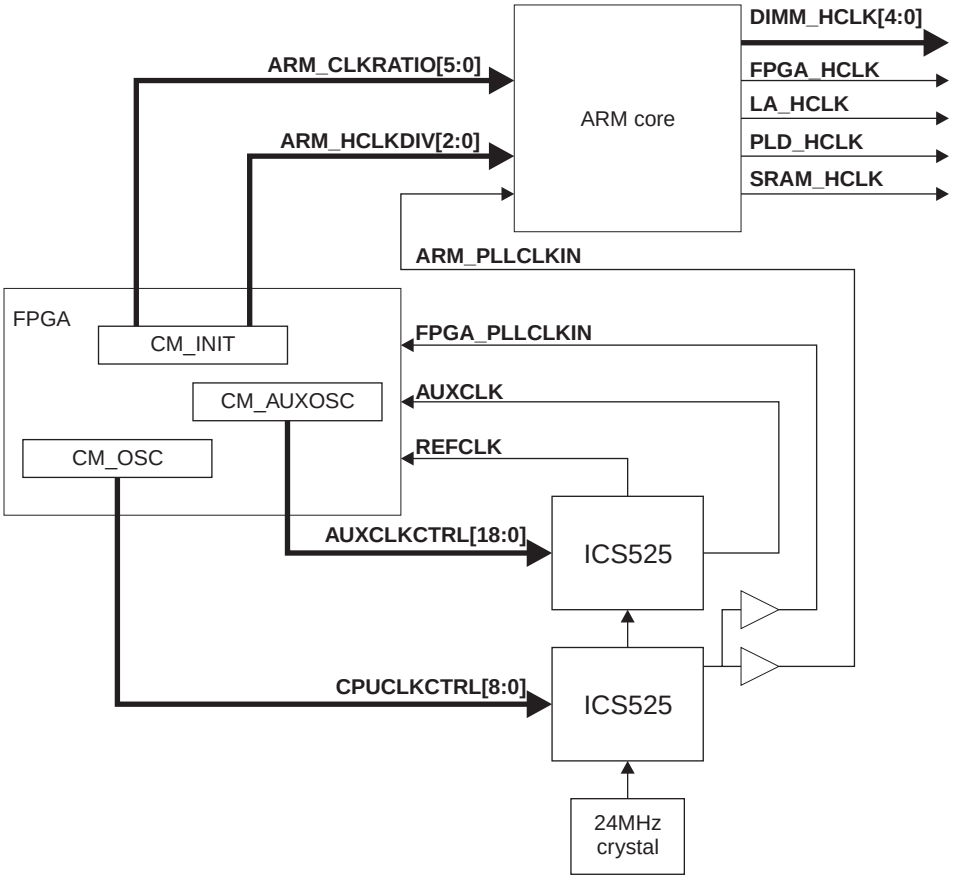


Figure 3-9 Core module clock generator

Copyright © 2000 ARM Limited. All rights reserved.

3.8.1 Clock generation functional overview

The ICS525s are supplied with a reference clock by a 24MHz crystal oscillator. The frequency of the outputs from the IC525s are controlled using their *divider* input pins. This enables them to produce a wide range of frequencies.

The output **ARM_PLLCLKIN** from the first ICS525 provides the clock source for a PLL incorporated into the ARM test chip. The PLL generates the core clock **CLK** which is internally divided to produce the AMBA AHB clock **HCLK**. The **HCLK** signal is distributed to devices around the core module using a number of dedicated output pins that are provided by the ARM test chip.

The signal **ARM_FPGACKIN** signal is supplied to the FPGA for future use. It always operates at the same frequency as **ARM_PLLCKIN**.

The second ICS525 generates the clock signal **AUXCLK** signal which is supplied to the FPGA for future use.

The fixed frequency 24MHz clock signal **REFCLK** is supplied to the FPGA for use by the internal counter CM_REFCNT. This can be used, for example, to provide real-time delays.

3.8.2 Programming the clocks

The frequency of clocks from an ICS525 is set by placing HIGH and LOW logic levels on its reference divider pins, *Voltage Controlled Oscillator* (VCO) divider pins, and output divider pins. These control the value of three parameters used to determine the output of the IC525. You can calculate the frequency using the formula:

freq = 48((VDW+ 8)/(RDW+2)*OD)

where:

VDW is the VCO divider word

RDW is the reference divider word

OD is the output divider.

This formula is used to calculate the frequency of **AUXCLK** but can be simplified for **ARM_PLLCKIN**, as described in *Setting the frequency of ARM_PLLCLKIN* on page 3-22.

Note

You can also calculate values for RDW, VDW, and OD using the ICS525 calculator on the Microclock website.

Copyright © 2000 ARM Limited. All rights reserved.

Setting the frequency of ARM_PLLCLKIN

In the case of the clock generator for **ARM_PLLCLKIN**, the values for RDW and OD are fixed by pull-up and pull-down resistors. VDW is programmable in the CM_OSC register (see *Core module oscillator register* on page 4-14). You can program the clock to frequencies in the range 12 to 160MHz in 1MHz steps. The value for RDW is fixed at 22 and the value for OD is fixed at 2.

This means that you can calculate the frequency of **ARM_PLLCKIN** from the following formula:

freq = VDW+ 8

where VDW can be assigned any value between a minimum of 4 and a maximum of 152.

Setting the frequency of AUXCLK

In the case of the clock generator for **AUXCLK**, the values for RDW, VDW, and OD are all programmable in the CM_AUXOSC register (see *Core module auxiliary oscillator register* on page 4-18). This gives frequencies in the range 1 to 160MHz with a better than 0.1% accuracy. The default frequency following a power-on reset is 32.369MHz.

3.8.3 Test chip clocks

The ARM9x6 test chip produces two clocks:

- the core clock **CLK**
- the local memory bus clock **HCLK**.

Core clock

The frequency of **CLK** is controlled by a PLL within the test chip. Test chips use the **ARM_CLKRATIO[5:0]** signals and, in some cases, one or more of the **USERIN[7:0]** signals as multipliers or dividers to determine the frequency of **CLK** (although actual usage of these signals is partner specific). These signals are programmed in the register CM_INIT (see *Core module initialization register* on page 4-22).

The internal PLL can be bypassed by setting PLLBYPASS bit in the CM_INIT register. This sets **CLK** to the same frequency as the reference clock and allows you to model implementations where the core does not have a PLL.

 Copyright © 2000 ARM Limited. All rights reserved.

Local memory bus clock

The test chip produces the **HCLK** signal by dividing **CLK** by a value of between 2 and 8. The divider can be effectively bypassed by setting a divide by value of 1. Division is controlled by the signals **ARM_HCLKDIV[3:0]** supplied by the FPGA. These signals are programmed in the CM_INIT register (see *Core module initialization register* on page 4-22). The **HCLK** signal is distributed by the test chip on four identical signals:

- **ARM_HCLK[0]** drives the FPGA
- **ARM_HCLK[1]** drives the SSRAM PLD
- **ARM_HCLK[2]** drives the SSRAM
- **ARM_HCLK[3]** drives a low-skew clock buffer which produces six outputs. These are supplied to the logic analyzer module, a test point, and to the SDRAM DIMM.

Copyright © 2000 ARM Limited. All rights reserved.

3.9 Multi-ICE support

The core module provides support for debug using JTAG. It provides a Multi-ICE connector and JTAG scan paths around the system. Figure 3-10 shows the Multi-ICE connector, the CONFIG link, and LED.

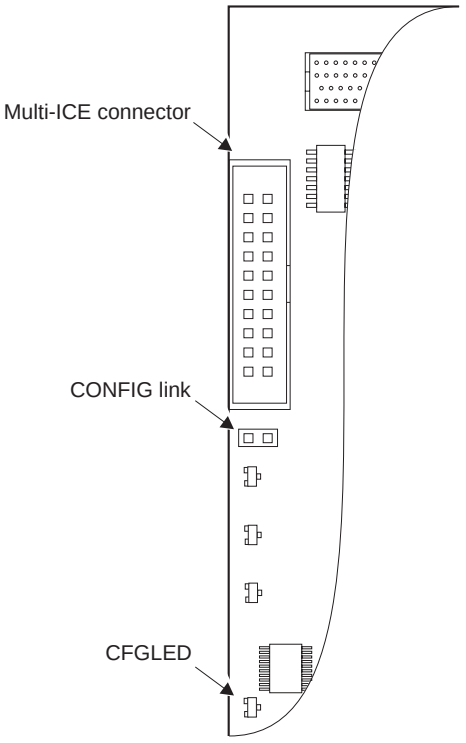


Figure 3-10 JTAG connector, CONFIG link, and LED

The CONFIG link is used to enable in-circuit programming of the FPGA and PLDs using Multi-ICE (see *JTAG connection modes* on page 3-27).

The Multi-ICE connector provides a set of JTAG signals allowing ARM and third-party JTAG debugging equipment to be used (see *JTAG signals* on page 3-28). If you are debugging a development system with multiple core modules, connect the JTAG debug hardware to the top core module.

Copyright © 2000 ARM Limited. All rights reserved.

3.9.1 JTAG scan paths

This section describes the routing of the JTAG scan chain on the core module. It discusses the data path and the clock path.

Data path

Figure 3-11 shows a simplified diagram of the data path.

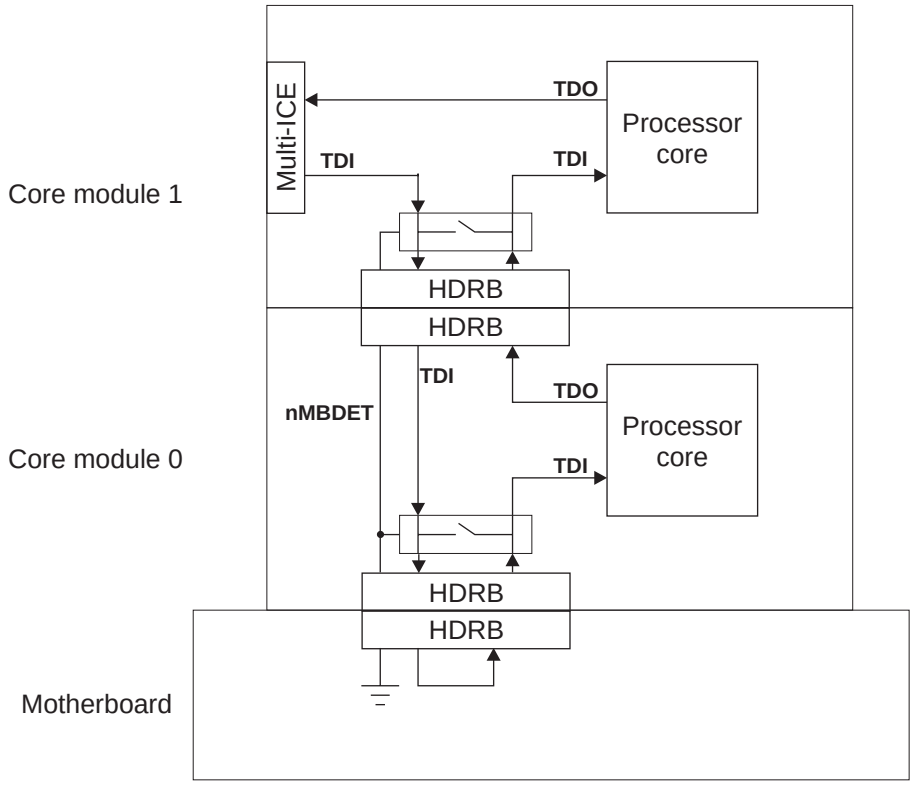


Figure 3-11 JTAG data path

When you use the core module as a standalone development system, the data path is routed to the processor core and back to the Multi-ICE connector.

If the core module is attached to an Integrator motherboard, the **TDI** signal from the top core module is routed down through the HDRB connectors of any modules in the stack to the motherboard. From there the path is routed back up the stack through each core

Copyright © 2000 ARM Limited. All rights reserved.

module, before being returned to the Multi-ICE connector as **TDO**. The motherboard detect signal **nMBDET** controls a switching circuit on the core module and, therefore, the routing of **TDI**.

The PLDs and FPGAs are included in the scan chain if the core module is in configuration mode, as described in *JTAG connection modes* on page 3-27.

Clock path

The clock path is routed in a similar way to the data path, although in the opposite direction. Figure 3-12 shows a simplified diagram of the clock path.

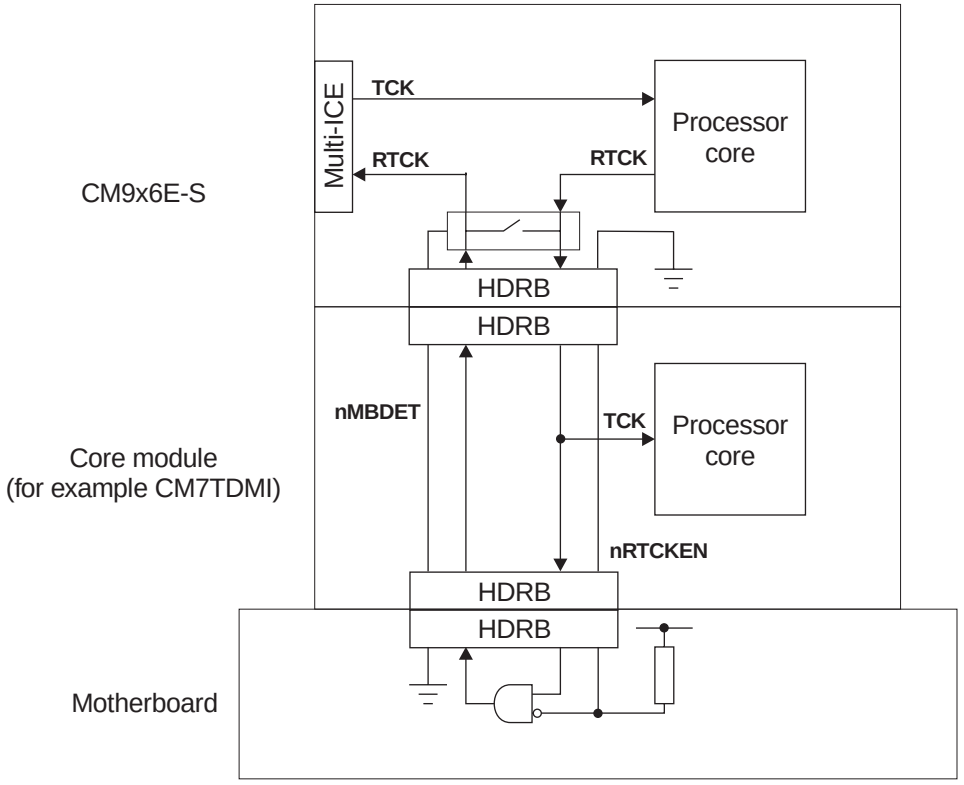


Figure 3-12 JTAG clock path

Copyright © 2000 ARM Limited. All rights reserved.

The ARM9x6E-S uses a synthesized core that samples **TCK**. This introduces a delay into the clock path. For this reason, it passes on the clock signal as **RTCK**, which is fed to the **TCK** input of the next device in the chain. The **RTCK** signal at the Multi-ICE connector is used by Multi-ICE to regulate the advance of **TCK**, a mechanism called adaptive clocking (see the *ARM Multi-ICE User Guide*).

The routing of the **TCK**/**RTCK** signals through the stack is controlled by switches, in a similar way to the data path, although in this case the loopback is controlled by the signal **nRTCKEN** and an AND gate on the motherboard (the pullups on **nMBDET** are omitted for clarity).

Not all cores need to sample **TCK** (for example, the ARM7TDMI) in which case the **TCK** signal is routed straight through to the next board down the stack. If one or more modules in a stack drives **RTCK** (and so asserts **nRTCKEN**), you must ensure that the board at the bottom of the stack provides the necessary return path. All Integrator motherboards do so.

3.9.2 JTAG connection modes

The core module operates in one of two JTAG modes:

- Normal mode
- Configuration mode.

Normal mode

Normal mode is selected by default (when a jumper is *not* fitted at the CONFIG link, see Figure 3-10 on page 3-24). It is the mode used for general system development and debug, including using trace (see *Embedded Trace support* on page 3-33). In this mode, only the processor core and debuggable devices on other modules are accessible on the scan chain, as shown in Figure 3-11 on page 3-25.

Configuration mode

In configuration mode, all FPGAs and PLDs in the system are added into the scan chain. This mode allows the programmable devices in the system to be reprogrammed in the field using Multi-ICE.

To select configuration mode, fit a jumper to the CONFIG link on the core module *at the top of the stack (*see Figure 3-10 on page 3-24*)*. This has the effect of pulling the **nCFGEN** signal LOW which illuminates the CFG LED (yellow) on each module in the stack and reroutes the JTAG scan path on all modules in the stack. The LED provides a warning that the development system is in the configuration mode.

Copyright © 2000 ARM Limited. All rights reserved.

Note

Configuration mode is guaranteed for a single core module attached to a motherboard but may be unreliable if more than one core module is attached.

After configuration or code updates you must:

1. Remove the CONFIG link.
2. Power cycle the development system.

The configuration mode allows FPGA and PLD code to be updated as follows:

- The FPGAs are volatile and load their configuration information from flash memory. The flash memory does not have a JTAG port, but can be programmed by loading designs into the FPGAs and PLDs that handle the transfer of data to the flash using JTAG.
- The PLDs are nonvolatile devices that can be programmed directly by JTAG.

3.9.3 JTAG signals

Figure 3-13 on page 3-29 shows the pinout of the Multi-ICE connector and Table 3-6 on page 3-29 provides a description of the JTAG signals.

Note

In the description in Table 3-6 on page 3-29, the term JTAG equipment refers to any hardware that can drive the JTAG signals to devices in the scan chain. Typically, Multi-ICE is used, although you can also use hardware from third-party suppliers to debug ARM processors.

 Copyright © 2000 ARM Limited. All rights reserved.

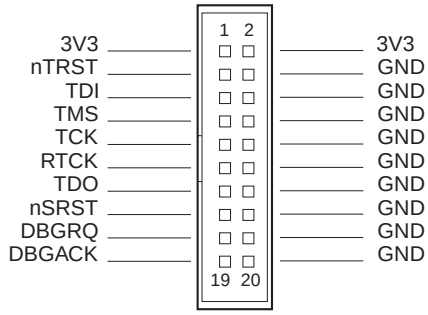


Figure 3-13 Multi-ICE connector pinout

Table 3-6 JTAG signal description

Name	Description	Function
DBGRQ	Debug request (from JTAG equipment)	**DBGRQ** is a request for the processor core to enter the debug state. It is provided for compatibility with third-party JTAG equipment.
DBGACK	Debug acknowledge (to JTAG equipment)	**DBGACK** indicates to the debugger that the processor core has entered debug mode. It is provided for compatibility with third-party JTAG equipment.
DONE	FPGA configured	**DONE** is an open-collector signal which indicates when FPGA configuration is complete. Although this signal is not a JTAG signal, it does effect **nSRST**. The **DONE** signal is routed between all FPGAs in the system through the HDRB connectors. The master reset controller on the motherboard senses this signal and holds all the boards in reset (by driving **nSRST** LOW) until all FPGAs are configured.
nCFGEN	Configuration enable (from jumper on module at the top of the stack)	**nCFGEN** is an active LOW signal used to put the boards into configuration mode. In configuration mode all FPGAs and PLDs are connected to the scan chain so that they can be configured by the JTAG equipment.
nRTCKEN	Return **TCK** enable (from core module to motherboard)	**nRTCKEN** is an active LOW signal driven by any core module that requires **RTCK** to be routed back to the JTAG equipment. If **nRTCKEN** is HIGH, the motherboard drives **RTCK** LOW. If **nRTCKEN** is LOW, the motherboard drives the **TCK** signal back up the stack to the JTAG equipment.

Copyright © 2000 ARM Limited. All rights reserved.

Table 3-6 JTAG signal description (continued)

Name	Description	Function
nSRST	System reset (bidirectional)	**nSRST** is an active LOW open-collector signal which can be driven by the JTAG equipment to reset the target board. Some JTAG equipment senses this line to determine when a board has been reset by the user. The open collector **nSRST** reset signal can be driven LOW by the reset controller on the core module to cause the motherboard to reset the whole system by driving **nSYSRST** LOW. This is also used in configuration mode to control the initialization pin (**nINIT**) on the FPGAs. Though not a JTAG signal, **nSRST** is described because it can be controlled by JTAG equipment.
nTRST	Test reset (from JTAG equipment)	This active LOW open-collector signal is used to reset the JTAG port and the associated debug circuitry on the processor. It is asserted at power-up by each module, and can be driven by the JTAG equipment. This signal is also used in configuration mode to control the programming pin (**nPROG**) on FPGAs.
RTCK	Return TCK (to JTAG equipment)	Some devices sample **TCK** (for example a synthesizable core with only one clock), and this has the effect of delaying the time at which a component actually captures data. **RTCK** is a mechanism for returning the sampled clock to the JTAG equipment, so that the clock is not advanced until the synchronizing device captured the data. In *adaptive clocking mode*, Multi-ICE is required to detect an edge on **RTCK** before changing **TCK**. In a multiple device JTAG chain, the **RTCK** output from a component connects to the **TCK** input of the down-stream device. The **RTCK** signal on the module connectors HDRB returns **TCK** to the JTAG equipment. If there are no synchronizing components in the scan chain then it is unnecessary to use the **RTCK** signal and it is connected to ground on the motherboard.
TCK	Test clock (from JTAG equipment)	**TCK** synchronizes all JTAG transactions. **TCK** connects to all JTAG components in the scan chain. Series termination resistors are used to reduce reflections and maintain good signal integrity. **TCK** flows down the stack of modules and connects to each JTAG component. However, if there is a device in the scan chain that synchronizes **TCK** to some other clock, then all down-stream devices are connected to the **RTCK** signal on that component (see **RTCK**).

Copyright © 2000 ARM Limited. All rights reserved.

Table 3-6 JTAG signal description (continued)

Name	Description	Function
TDI	Test data in (from JTAG equipment)	**TDI** goes down the stack of modules to the motherboard and then back up the stack, labeled **TDO**, connecting to each component in the scan chain.
TDO	Test data out (to JTAG equipment)	**TDO** is the return path of the data input signal **TDI**. The module connectors HDRB have two pins labeled **TDI** and **TDO**. **TDI** refers to data flowing down the stack and **TDO** to data flowing up the stack. The JTAG components are connected in the return path so that the length of track driven by the last component in the chain is kept as short as possible.
TMS	Test mode select (from JTAG equipment)	**TMS** controls transitions in the tap controller state machine. **TMS** connects to all JTAG components in the scan chain as the signal flows down the module stack.

Copyright © 2000 ARM Limited. All rights reserved.

3.9.4 Debug communications interrupts

The processor core incorporates EmbeddedICE logic which contains a communications channel used for passing information between the core and the JTAG equipment. The debug communications channel is implemented as coprocessor 14.

The processor accesses the debug communications channel registers using `MCR` and `MRC` instructions. The JTAG equipment reads and writes the register using the scan chain. For a description of the debug communications channel, see the *ARM966E-S Technical Reference Manual* or *ARM946E-S Technical Reference Manual*.

Interrupts can be used to signal when data has been written into one side of the register and is available for reading from the other side. These interrupts are supported by the interrupt controller within the core module FPGA, and can be enabled and cleared by accessing the interrupt registers (see *Core module interrupt registers* on page 4-26).

Copyright © 2000 ARM Limited. All rights reserved.

3.10 Embedded Trace support

The ARM9x6E-S processors incorporate an *ARM9 Embedded Trace Macrocell* (ETM9). This enables you to carry out real-time debugging by connecting external trace equipment to the core module. To trace program flow, the ETM broadcasts branch addresses, data accesses, and status information through the trace port. Later in the debug process, the complete instruction flow can be reconstructed by the ARM *Trace Debug Tools* (TDT).

Note

It is not possible to reconstruct self-modifying code.

3.10.1 About using trace

Figure 3-14 illustrates a trace debugging setup with the core module.

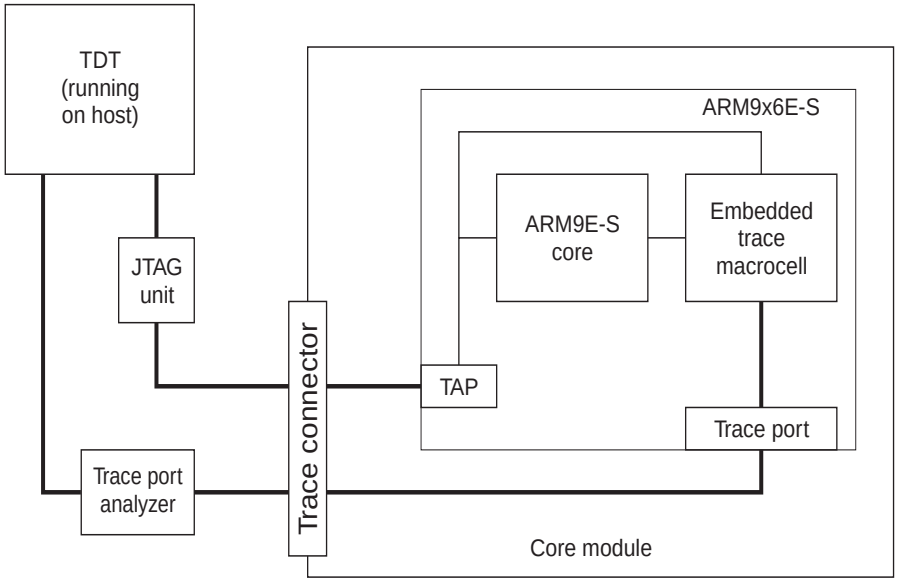


Figure 3-14 Trace connection

Note

The routing of the JTAG scan chain on the Integrator system is described in *Multi-ICE support* on page 3-24.

Copyright © 2000 ARM Limited. All rights reserved.

The components in the trace debug setup shown in Figure 3-14 are as follows:

Embedded trace macrocell

The ETM monitors the ARM core buses and outputs compressed information through the trace port to a trace port analyzer. The on-chip ETM contains trigger and filter logic to control what is traced.

Trace port analyzer

The *Trace Port Analyzer* (TPA) is an external device that stores information from the trace port.

JTAG unit

This is a protocol converter that converts debug commands from the debugger into JTAG messages for the ETM. The JTAG unit may be a separate device or may be incorporated within the TPA.

Trace debug tools

The *Trace Debug Tools* (TDT) is an optional component of the ARM Developer Suite that runs on a host system. It is used to set up the filter logic, retrieve data from the analyzer, and reconstruct an historical view of processor activity. For further information, see the *ADS Trace Debug Tools User Guide*.

Copyright © 2000 ARM Limited. All rights reserved.

3.10.2 Core module trace configuration

The ARM9x6E-S test chip provides a *large* ETM configuration. The resources provided by this configuration are summarized in Table 3-7.

Table 3-7 ETM resources

Resource type	Number
Address comparator pairs	8
Data comparator	8
Memory map decoders	16
Counters	4
Sequencer	Yes
External inputs	4
External outputs	4
FIFOFULL	Yes
FIFO depth	45
Trace packet width	4/8/16

The TRACE connector enables you to connect an external embedded trace interface module.

3.10.3 Trace interface description

The logic analyzer connection is a high-density AMP Mictor connector. The pinout for this connector is provided in Trace interface pinout on page A-9.

Copyright © 2000 ARM Limited. All rights reserved.

 Copyright © 2000 ARM Limited. All rights reserved.

Chapter 4
Programmer's Reference

This chapter describes the memory map and the status and control registers. It contains the following sections:

- *About the ARM9x6E-S memory map* on page 4-2
- *Core module memory map configuration* on page 4-4
- *Processor configuration* on page 4-11
- *Core module control registers* on page 4-12
- *Core module flag registers* on page 4-25
- *Core module interrupt registers* on page 4-26
- *SDRAM SPD memory* on page 4-30.

Copyright © 2000 ARM Limited. All rights reserved.

4.1 About the ARM9x6E-S memory map

To understand the memory map options provided by the Integrator/CM9x6E-S, it is helpful to be familiar with the main features of the ARM946E-S and ARM966E-S that influence their memory maps. This section provides a brief overview of these features. For detailed information, see the *ARM946E-S Technical Reference Manual* or *ARM966E-S Technical Reference Manual*.

The ARM946E-S and ARM966E-S both incorporate internal *Tightly-Coupled RAM* (TCRAM), and both incorporate a write buffer. However, the two processors differ in the following ways:

- The ARM946E-S incorporates a protection unit that allows you to define protection attributes for up to eight regions. Protection attributes include cacheable, bufferable, user access, and supervisor access.
- By contrast, the ARM966E-S has sixteen fixed attribute regions within its memory map that are predefined alternately as buffered and unbuffered.
- The ARM946E-S has separate areas of internal TCRAM for data and instructions. These can be individually enabled or disabled, and can be varied in size. The Instruction TCRAM (I-RAM), if enabled, is always located at address `0x00000000`, but the Data TCRAM (D-RAM), if enabled, can be assigned to any size-aligned boundary.
- The ARM966E-S also has areas for internal I-RAM and D-RAM that can be separately enabled or disabled but differs in that these are of fixed size. The I-RAM, if enabled, is always located at base address `0x00000000` and the D-RAM, if enabled, is always located at base address `0x04000000`.

Figure 4-1 on page 4-3 shows example memory maps for both the ARM946E-S and ARM966E-S. The example for both cores shows the memory map with I-RAM and D-RAM enabled. Accesses within these areas stay on-chip. This means, for example, that any Integrator memory at address `0x00000000` - `0x04000000` in the ARM966E-S memory map is masked if the I-RAM is enabled. This means, also, that you should be careful when choosing a location for the D-RAM on the ARM946E-S that you do not mask registers or other important locations.

On all ARM946E-S and ARM966E-S processors, the TCRAMs are enabled using the system coprocessor register (CP15r1). In some of these processors this bit is initialized at reset from the INITRAM bit (see *Core module initialization register* on page 4-22).

In the ARM966E-S, both I-RAM and D-RAM are enabled or disabled immediately after reset according to the state of INITRAM. In the ARM946E-S rev1 the I-RAM is enabled or disabled immediately after reset according to the state of INITRAM. In the ARM946E-S rev0, both I-RAM and D-RAM are disabled after reset.

 Copyright © 2000 ARM Limited. All rights reserved.

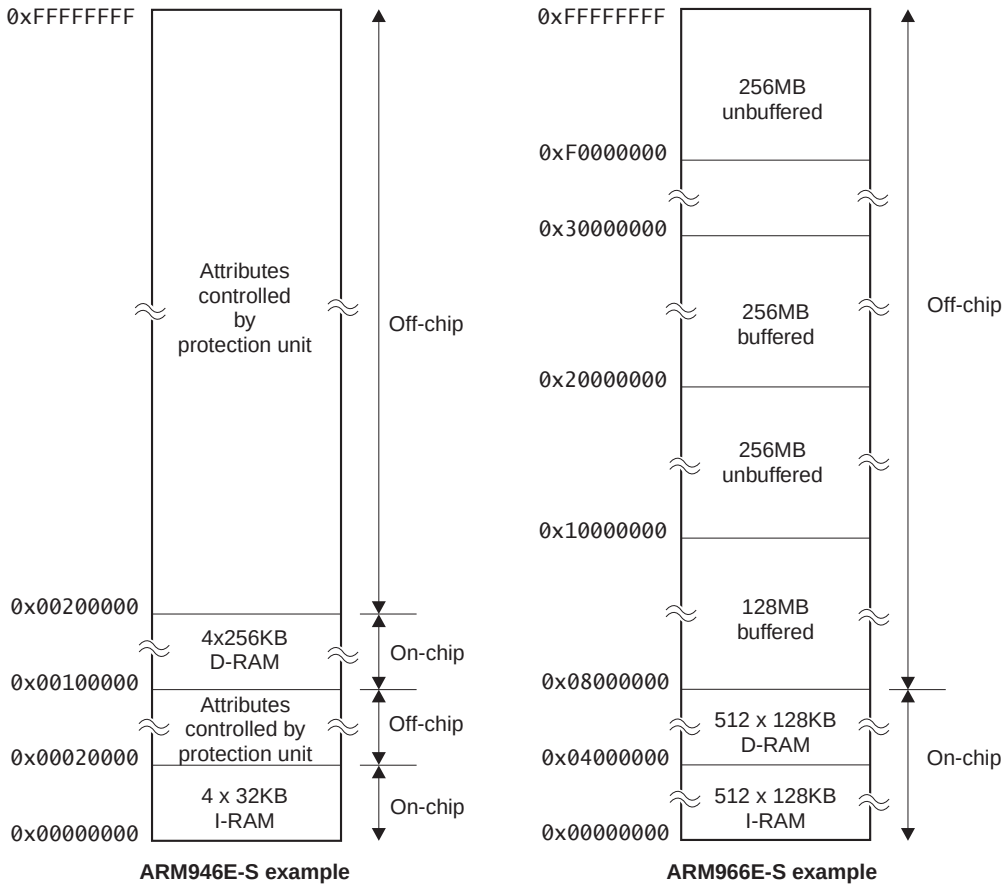


Figure 4-1 Example ARM946E-S and ARM966E-S memory maps

Typically, the amount of I-RAM and D-RAM physically present within the test chip is less than the address space allocated to it. However, each is aliased throughout its assigned range.

Note

A feature of the Integrator/CM9x6E-S core modules is that it allows you to use the on-board SSRAM to mimic the TCRAM (see *SSRAM mode selection* on page 4-6). This enables you to model different TCRAM implementations, even if your test chip does not have any TCRAM, and provides logic analyzer visibility of accesses to the RAM. However, if you use the SSRAM to mimic the TCRAM, performance will be slower.

Copyright © 2000 ARM Limited. All rights reserved.

4.2 Core module memory map configuration

The core module provides a configurable memory map that enables you to experiment with different system configurations. The mapping of the control and status registers, and region for access to motherboard resources are fixed. However, the mapping for the SDRAM, SSRAM and TCRAM (built into the test chip) change when the configuration of the core module is changed. These configuration options are described in the following sections:

- *SSRAM mode selection* on page 4-6
- *Using REMAP* on page 4-7.

Table 4-1 shows the configurable area of the memory map and the fixed areas.

Table 4-1 Overview of core module memory map

Address range	Size	Description
`0x00000000` to `0x07FFFFFF`	128MB	Configurable mapping of: TCRAM On-board SSRAM (alias) boot ROM (on motherboard) flash (on motherboard) SDRAM
`0x08000000` to `0x0FFFFFFF`	128MB	SDRAM
`0x10000000` to `0x107FFFFF`	8MB	Core module registers
`0x10800000` to `0x10FFFFFF`	8MB	On-board SSRAM (alias)
`0x11000000` to `0xFFFFFFFF`	3824MB	System bus (Abort if core module is not fitted to a motherboard)

Configuration of the memory map in the address space `0x00000000` to `0x07FFFFFF` is controlled by the SSRAM mode and by REMAP. Figure 4-2 on page 4-5 shows the affect on the memory map of configuring the SSRAM mode and REMAP.

Note

Figure 4-2 on page 4-5 shows the memory map for the four SSRAM-mode/REMAP configurations with the TCRAM *disabled*. The TCRAM, when enabled, masks accesses to the areas of the memory map that it occupies (see *About the ARM9x6E-S memory map* on page 4-2.)

Copyright © 2000 ARM Limited. All rights reserved.

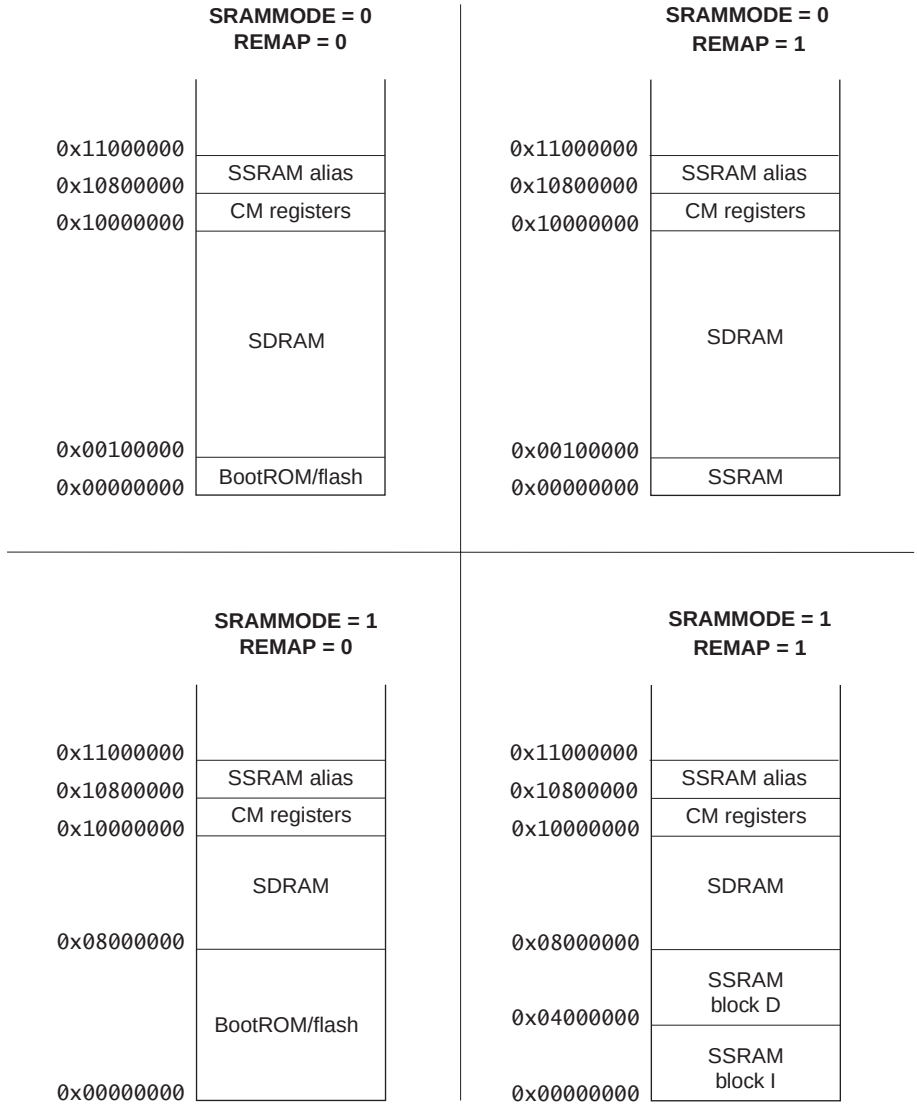


Figure 4-2 Configurable map

Copyright © 2000 ARM Limited. All rights reserved.

4.2.1 SSRAM mode selection

The on-board SSRAM appears within the configurable region and at an alias location (see *SSRAM alias* on page 4-8). Within the configurable region, you can map the SSRAM either in place of the TCRAM or in a similar way to previous Integrator core modules. This mapping is defined by the operating mode of the SSRAM as follows:

SSRAM Mode 0

This mode is similar to other Integrator core modules. The SSRAM appears as a 1MB block at `0x00000000` to `0x000FFFFF`. If the core module is fitted to a motherboard and the REMAP bit is 0, then accesses within `0x00000000` to `0x000FFFFF` are routed to the boot ROM or flash on the motherboard (see *Using REMAP* on page 4-7).

SSRAM Mode 1

This mode uses the on-board SSRAM to emulate the TCRAM. The SSRAM is organized as two 512KB blocks. The first of these (block I) is accessed within the address space `0x00000000` to `0x03FFFFFF` and the second (block D) within the address space `0x04000000` to `0x07FFFFFF`. Each of these 64MB regions is filled with 128 aliases of the associated SSRAM block.

The two modes of operations are selected using the SRAMMODE bit in the CM_INIT register (see *Core module initialization register* on page 4-22).

Changing the SSRAM mode also affects the amount of SDRAM that can be accessed. The SDRAM always appears contiguously at the top of the SSRAM. Table 4-2 shows how the configurable area of the memory map appears for the two SSRAM modes.

Table 4-2 Effect of SSRAM mode 0 and mode 1

SRAMMODE	Address range	Size	Description
0	`0x00000000` to `0x000FFFFF`	1MB	SSRAM
0	`0x00100000` to `0x0FFFFFFF`	255MB	SDRAM
1	`0x00000000` to `0x03FFFFFF`	64MB	SSRAM block I (128 copies of block I)
1	`0x04000000` to `0x07FFFFFF`	64MB	SSRAM block D (128 copies of block D)
1	`0x08000000` to `0x0FFFFFFF`	128MB	SDRAM

Copyright © 2000 ARM Limited. All rights reserved.

4.2.2 Using REMAP

When the core module is mounted on an Integrator motherboard, the mapping of the boot ROM on the motherboard or the SSRAM can be changed using the REMAP bit in the CM_CTRL register (see *Core module control register* on page 4-15). There is an interaction between the REMAP bit and the SSRAM mode. This interaction is illustrated in Figure 4-2 on page 4-5.

The REMAP bit functions as follows:

REMAP=0 In SSRAM mode 0, the boot ROM or flash on the motherboard is mapped into the address range `0x00000000` to `0x000FFFFF`. The SDRAM is accessible from `0x00100000`.

Note

You can set REMAP to 0 only if the core module is attached to a motherboard.

In SSRAM mode 1, the boot ROM or flash on the motherboard is multiply mapped into the address range `0x00000000` to `0x07FFFFFF`. The SDRAM is accessible from `0x08000000` to `0x0FFFFFFF`.

REMAP=1 In SSRAM mode 0, the SSRAM is mapped to the address range `0x00000000` to `0x000FFFFF` (as one 1MB block).

In SSRAM mode 1, the first block (block I) of SSRAM is multiply mapped into the address range `0x00000000` to `0x03FFFFFF` and the second block (block D) of SSRAM is multiply mapped into `0x04000000` to `0x07FFFFFF`.

Note

Program execution normally begins at `0x00000000`. A switch on the motherboard determines whether the boot ROM or flash is mapped to this location, enabling you to boot from the boot ROM or from flash memory. Refer to the user guide for your motherboard for more information.

Copyright © 2000 ARM Limited. All rights reserved.

4.3 SSRAM alias

The SSRAM is always accessible at `0x10800000` to `0x10FFFFFF`, independently of whether the TCRAM is enabled or disabled, and of the SRAMMODE, and REMAP controls. The SSRAM is mapped repeatedly to fill the SSRAM alias region. However, the mapping of each SSRAM repeat depends on the SSRAM mode in operation. In SSRAM mode 0, each repeat contains the whole 1MB of SSRAM as one block. In SSRAM mode 1, each repeat contains a 512KB alias of block I and a 512KB alias of block D, (see *SSRAM mode selection* on page 4-6).

The mapping of the aliases for the two SSRAM modes is shown in Figure 4-3.

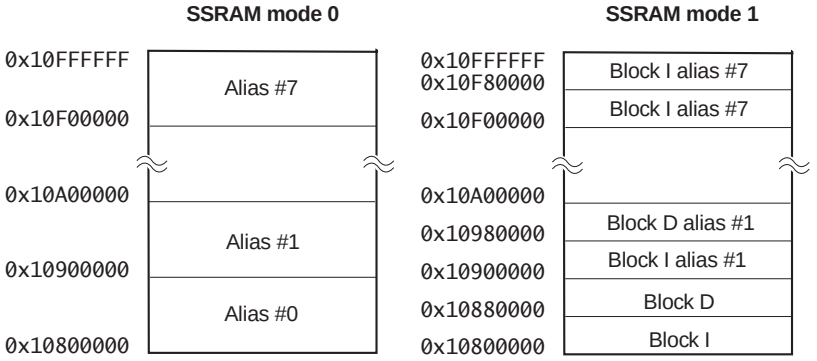


Figure 4-3 SSRAM aliases

Copyright © 2000 ARM Limited. All rights reserved.

4.4 SDRAM mapping

The Integrator memory map provides two regions in which SDRAM can be accessed. One region provides access by the local processor, and the second allows access by any master within the Integrator system. The global access region is not available on a standalone core module.

4.4.1 Local access

The amount of SDRAM that can be accessed within this region depends upon the SSRAM mode. Accesses to the SDRAM in this region are restricted to the core on the same core module. In SSRAM mode 0, the SDRAM is mapped to a 255MB space at `0x00100000` to `0x0FFFFFFF`. In SSRAM mode 1, the SDRAM is mapped to a 128MB space at `0x08000000` to `0x0FFFFFFF`.

You can fit SDRAM DIMMs of up to 256MB to the core module but you cannot access the lower 1MB (SSRAM mode 0) or 128MB (SSRAM mode 1) of the DIMM within the local access region. The SDRAM DIMM is mapped repeatedly within the accessible space but the SSRAM masks part of the SDRAM local access region. This means, for example, that in SSRAM mode 1, the first eight aliases of a 16MB DIMM are masked, whereas in SSRAM mode 0, only 1MB of the first alias is masked.

Figure 4-4 shows the repeat mapping of a 64MB SDRAM with SSRAM mode 0 in operation. The lowest image is partly masked by the SSRAM, but the repeat images are fully accessible.

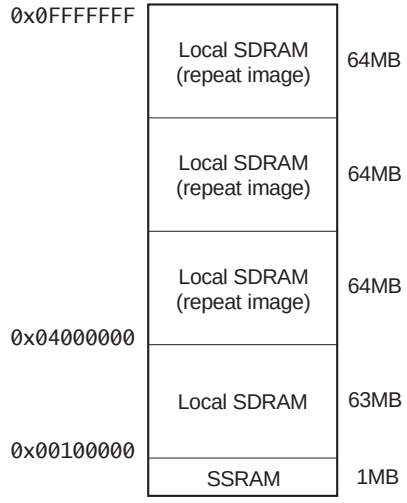


Figure 4-4 Repeat DRAM mapping

Copyright © 2000 ARM Limited. All rights reserved.

4.4.2 Global SDRAM access

If the core module is mounted on a motherboard, the SDRAM appears within a 256MB space in the *alias* memory region of the overall Integrator system memory map (see the user guide for your motherboard). The SDRAM can be accessed by all bus masters within this region.

The alias address for a core module SDRAM is automatically controlled by its position in the stack (see *Module ID selection* on page 3-18). Figure 4-5 shows the alias address of the SDRAM on four core modules.

A processor can determine which core module it is on and, therefore, the alias location of its own SDRAM by reading the CM_STAT register, (see *Core module status register* on page 4-16).

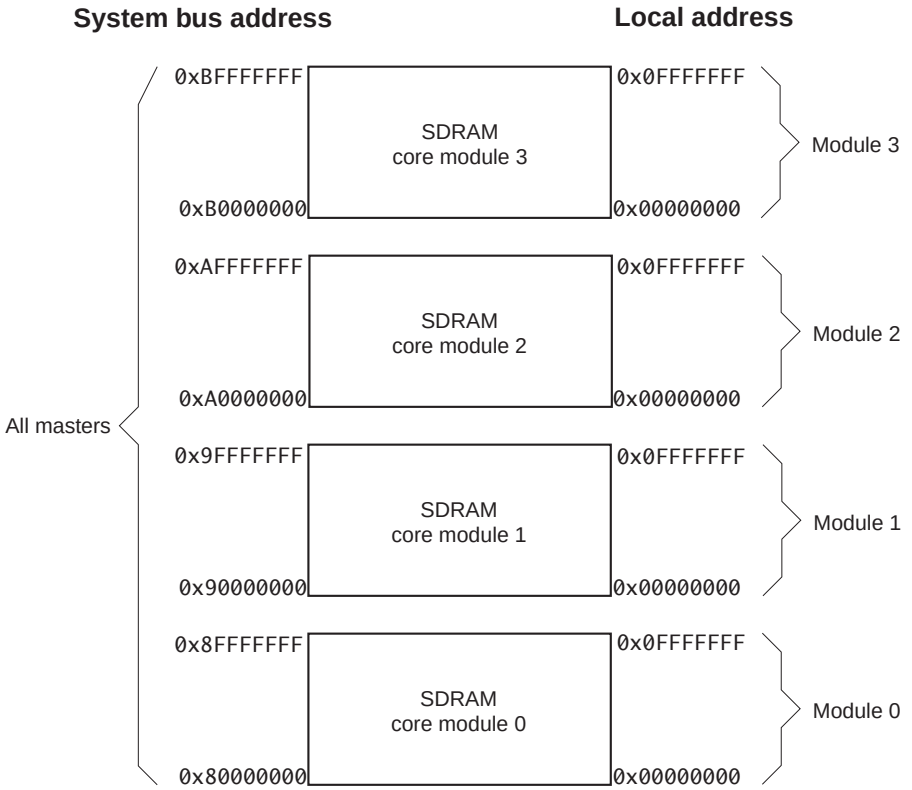


Figure 4-5 Core module local and alias addresses

Copyright © 2000 ARM Limited. All rights reserved.

4.5 Processor configuration

The ARM9x6E-S processors provide several configuration options. Configuration options include:

- TCRAM
- vector locations
- byte order (big or little-endian).

These options are described in *Test chip configuration control* on page 3-2.

4.5.1 Limitations

The core module and Integrator motherboards have no physical memory at the high vector location `0xFFFF0000`. When the core module is being used with a motherboard, it is possible to implement an area of physical memory at this address on a logic module.

The Integrator/CM9x6E-S core modules do not support big-endian byte ordering.

4.5.2 Identifying the test chip type on your core module

To identify the manufacturer and type of ARM core fitted to your core module, read the SI_ID bits in the CM_STAT registers (see *Core module status register* on page 4-16). These bits are set by resistors that are fitted to the board at build time. It is important to be able to identify the type of core fitted because different manufacturers implement different features in the test chip. These differences typically relate to the control signals for the PLL clock dividers, to the size of TCRAM, and to the cache sizes.

Copyright © 2000 ARM Limited. All rights reserved.

4.6 Core module control registers

The core module status and control registers allow the processor to determine its environment and to control core module operations. The registers, listed in Table 4-3, are located at `0x10000000` and can only be accessed by the local processor.

Table 4-3 Core module status, control, and interrupt registers

Register Name	Address	Access	Reset	Description
CM_ID	`0x10000000`	R	Static	Core module identification register
CM_PROC	`0x10000004`	R	Static	Core module processor register
CM_OSC	`0x10000008`	R/W	POR	Core module oscillator register
CM_CTRL	`0x1000000C`	R/W	Reset	Core module control
CM_STAT	`0x10000010`	R	Reset	Core module status
CM_LOCK	`0x10000014`	R/W	Reset	Core module lock
CM_LMBUSCNT	`0x10000018`	R	Reset	Core module local memory bus cycle counter
CM_AUXOSC	`0x1000001C`	R/W	POR	Core module auxiliary clock oscillator register
CM_SDRAM	`0x10000020`	R/W	POR	SDRAM status and control register
CM_INIT	`0x10000024`	R/W	POR	Core module initialization register
CM_REFCT	`0x10000028`	R	Reset	Reference clock cycle counter
CM_UNUSED1	`0x1000002C`	-	-	Reserved
CM_FLAGS	`0x10000030`	R	Reset	Core module flag register
CM_FLAGSS	`0x10000030`	W	Reset	Core module flag set register
CM_FLAGSC	`0x10000034`	W	Reset	Core module flag clear register
CM_NVFLAGS	`0x10000038`	R	POR	Core module non-volatile flag register
CM_NVFLAGSS	`0x10000038`	W	POR	Core module non-volatile flag set register
CM_NVFLAGSC	`0x1000003C`	W	POR	Core module non-volatile flag clear register
CM_IRQ_STATUS	`0x10000040`	R	Reset	Core module IRQ status register
CM_IRQ_RSTAT	`0x10000044`	R	Reset	Core module IRQ raw status register

Copyright © 2000 ARM Limited. All rights reserved.

Table 4-3 Core module status, control, and interrupt registers

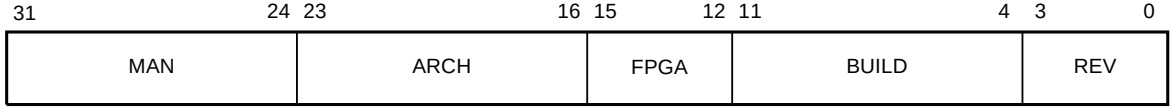
Register Name	Address	Access	Reset	Description
CM_IRQ_ENSET	0x10000048	R	Reset	Core module IRQ enable set register
CM_IRQ_ENCLR	0x1000004C	W	Reset	Core module IRQ enable clear register
CM_SOFT_INTSET	0x10000050	R/W	Reset	Core module software interrupt set
CM_SOFT_INTCLR	0x10000054	W	Reset	Core module software interrupt clear
CM_FIQ_STATUS	0x10000060	R	Reset	Core module FIQ status register
CM_FIQ_RSTAT	0x10000064	R	Reset	Core module FIQ raw status register
CM_FIQ_ENSET	0x10000068	R/W	Reset	Core module FIQ enable set register
CM_FIQ_ENCLR	0x1000006C	W	Reset	Core module FIR enable clear register
CM_SPD	0x10000100 to 0x100001FC	R	POR	SDRAM SPD memory

Note

All registers are 32-bits wide and only support word-wide writes. Bits marked as *reserved* in the following sections should be preserved using read-modify-write operations.

4.6.1 Core module ID register

The core module ID register (CM_ID) is a read-only register that identifies the board manufacturer, board type, and revision.



Copyright © 2000 ARM Limited. All rights reserved.

Table 4-4 describes the core module ID register bits.

Table 4-4 CM_ID register bit descriptions

Bits	Name	Access	Function
31:24	MAN	Read	Manufacturer: 0x41 = ARM
23:16	ARCH	Read	Architecture: 0x1A = AHB interface, 64-bit SDRAM
15:12	FPGA	Read	FPGA type: 0x3 = XCV600
11:4	BUILD	Read	Build value (ARM internal use)
3:0	REV	Read	Revision: 0x0 = Rev A 0x1 = Rev B

4.6.2 Core module processor register

The core module processor register (CM_PROC) is a read-only register that contains the value 0x00000000. This is provided for compatibility with processors that do not have a system control coprocessor (CP15). For the ARM9x6E-S cores, information about the processor can be obtained by reading coprocessor 15 register 0 (CP15 c0).

4.6.3 Core module oscillator register

The core module oscillator register (CM_OSC) is a read/write register that controls the frequency of the clock generated by the clock generator for **ARM_PLLCLKIN** (see *Programming the clocks* on page 3-21).

Note

Before writing to the CM_OSC register, you must unlock it by writing the value 0x0000A05F to the CM_LOCK register. After writing the CM_OSC register, relock it by writing any value other than 0x0000A05F to the CM_LOCK register.

Copyright © 2000 ARM Limited. All rights reserved.

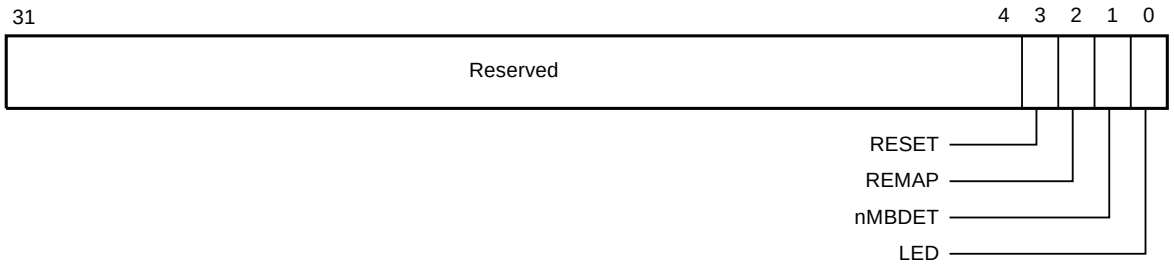
Table 4-5 describes the core module oscillator register bits.

Table 4-5 CM_OSC register

Bits	Name	Access	Function
31:25	Reserved	Use read-modify-write to preserve value.	
24:23	BMODE	Read	Bus mode. Always contains b10. Select the bus operation by writing to CM_INIT.
22:8	Reserved	Use read-modify-write to preserve value.	
7:0	PLL_VDW	Read/write	Core clock VCO divider word. Defines the binary value on the V[7:0] pins of the clock generator for ARM_PLLCKIN (V[8] is tied LOW). `b01110000` = 120MHz (default).

4.6.4 Core module control register

The core module control register (CM_CTRL) is a read/write register that provides control of a number of user-configurable features of the core module.



Copyright © 2000 ARM Limited. All rights reserved.

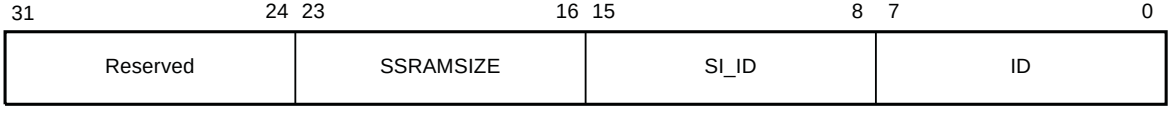
Table 4-6 describes the core module control register bits

Table 4-6 CM_CTRL register

Bits	Name	Access	Function
31:4	Reserved	Use read-modify-write to preserve value.	
3	RESET	Write	This is used to reset the core module, the motherboard on which it is mounted, and any modules in a stack. A reset is triggered by writing a 1. Reading this bit always returns a 0 allowing you to use read-modify-write operations without masking the RESET bit.
2	REMAP	Read/write	This only has affect when the core module is mounted on a motherboard. The function of remap is described in *Using REMAP* on page 4-7.
1	nMBDET	Read	This bit indicates whether or not the core module is mounted on a motherboard: 0 = mounted on motherboard 1 = standalone.
0	LED	Read/write	This bit controls the green MISC LED on the core module: 0 = LED OFF 1 = LED ON.

4.6.5 Core module status register

The core module status register (CM_STAT) is a read-only register that can be read to determine the size of the SSRAM present, the test chip type (see *Processor configuration* on page 4-11), and where in a stack this core module is positioned.



Copyright © 2000 ARM Limited. All rights reserved.

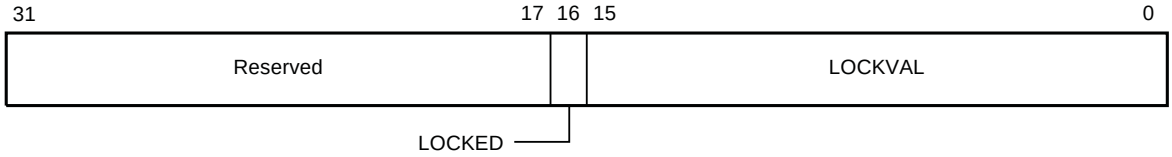
Table 4-7 describes the core module status register bits.

Table 4-7 CM_STAT register

Bit	Name	Access	Function
31:24	Reserved	Use read-modify-write to preserve value.	
23:16	SSRAMSIZE	Read	SSRAM size. This contains `0x10` on the CM9x6E-S to indicate that 1MB is fitted.
15:8	SI_ID	Read	Silicon manufacturer identification. Identifies the manufacturer and type of core fitted to the module: `0x00` = unknown or socket fitted `0x01` = Lucent (3V3 core and pads) `0x02` = LSI (G11 0.25µ – 2V5 core, 3V3 pads) `0x03` = LSI (G12 0.18µ – 1V8 core, 3V3 pads) `0x04` - `0xFF` = reserved.
7:0	ID	Read	Card number in stack: `0x00` = core module 0 `0x01` = core module 1 `0x02` = core module 2 `0x03` = core module 3 `0xFF` = invalid or no motherboard attached.

4.6.6 Core module lock register

The core module lock register (CM_LOCK) is a read/write register that is used to control access to the CM_OSC, CM-INIT, and CM_AUXOSC registers, allowing them to be locked and unlocked. This mechanism prevents these registers from being overwritten accidently.



Copyright © 2000 ARM Limited. All rights reserved.

Table 4-8 describes the core module lock register bits.

Table 4-8 CM_LOCK register

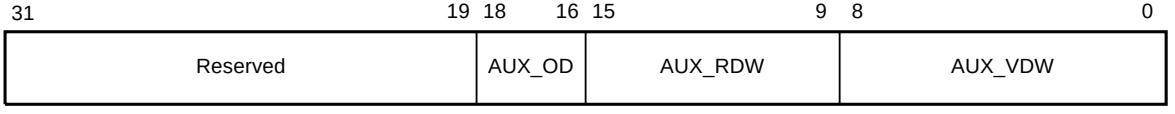
Bits	Name	Access	Function
31:17	Reserved	Use read-modify-write to preserve value.	
16	LOCKED	Read	This bit indicates if the CM_OSC register is locked or unlocked: 0 = unlocked 1 = locked.
15:0	LOCKVAL	Read/write	Write the value `0x0000A05F` to this register to enable write accesses to the CM_OSC register. Write any other value to this register to lock the CM_OSC register.

4.6.7 Core module local memory bus cycle counter

This register (CM_LMBUSCNT) provides a 32-bit count value. The count increments at the memory bus frequency and can be used as a cycle counter for performance measurement. The register is reset to zero by a reset.

4.6.8 Core module auxiliary oscillator register

The core module auxiliary oscillator register (CM_AUXOSC) is a read/write register that controls the frequency of the clock generated by the clock generator for **AUXCLK** (see *Programming the clocks* on page 3-21). This register enables you to set the all three of the control inputs to the clock generator. The default setting of this register gives a 32.369MHz output.



Note

Before writing to the CM_AUXOSC register, you must unlock it by writing the value `0x0000A05F` to the CM_LOCK register. After writing the CM_AUXOSC register, relock it by writing any value other than `0x0000A05F` to the CM_LOCK register.

Copyright © 2000 ARM Limited. All rights reserved.

Table 4-9 describes the core module oscillator register bits.

Table 4-9 CM_AUXOSC register

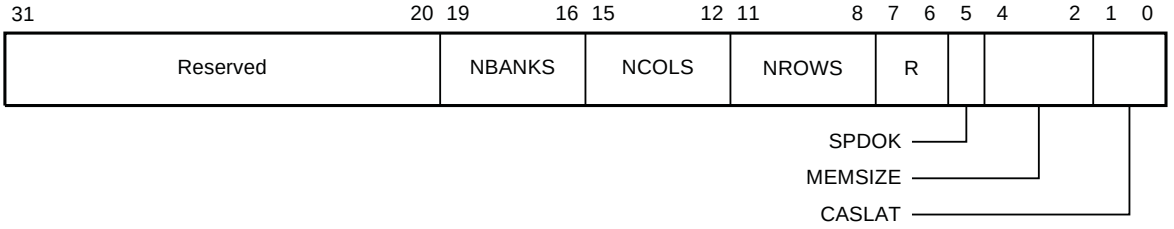
Bits	Name	Access	Function
31:19	Reserved	Use read-modify-write to preserve value.	
18:16	AUX_OD	Read-write	Auxiliary output divider. Sets the binary code on the S[2:0] pins of the clock generator. The divider is encoded as follows: 000 = divide by 0 001 = divide by 2 010 = divide by 8 011 = divide by 4 100 = divide by 5 101 = divide by 7 110 = divide by 9 111 = divide by 6 (default).
15:9	AUX_RDW	Read/write	Auxiliary reference divider word. Defines the binary value on the R[6:0] pins of the clock generator. `b0111111` = 63 (default).
8:0	AUX_VDW	Read/write	Auxiliary clock VCO divider word. Defines the binary value on the V[8:0] pins of the clock generator. `b011111111` = 255 (default).

Copyright © 2000 ARM Limited. All rights reserved.

4.6.9 SDRAM status and control register

The SDRAM status and control register (CM_SDRAM) is a read/write register used to set the configuration parameters for the SDRAM DIMM. This control is necessary because of the variety of module sizes and types available.

Writing a value to this register automatically updates the mode register on the SDRAM DIMM.



Note

Before the SDRAM is used it is necessary to read the SPD memory and program the CM_SDRAM register with the parameters indicated in Table 4-10. If these values are not correctly set then SDRAM accesses may be slow or unreliable. See *SDRAM SPD memory* on page 4-30.

Table 4-10 describes the SDRAM status and control register bits.

Table 4-10 CM_SDRAM register

Bits	Name	Access	Function
31:20	Reserved	Use read-modify-write to preserve value.	
19:16	NBANKS	Read/write	Number of SDRAM banks. Should be set to the same value as byte 5 of SPD EEPROM.
15:12	NCOLS	Read/write	Number of SDRAM columns. Should be set to the same value as byte 4 of SPD EEPROM.
11:8	NROWS	Read/write	Number of SDRAM rows. Should be set to the same value as byte 3 of SPD EEPROM.
7:6	Reserved	Use read-modify-write to preserve value.	

Copyright © 2000 ARM Limited. All rights reserved.

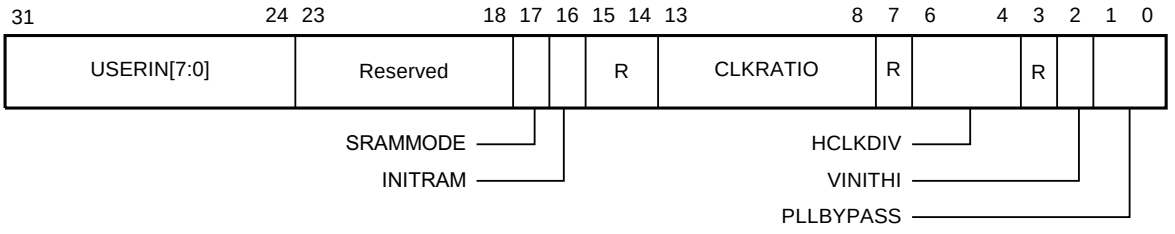
Table 4-10 CM_SDRAM register

Bits	Name	Access	Function
5	SPDOK	Read	This bit indicates that the automatic copying of the SPD data from the SDRAM module into CM_SPDMEM is complete: 1 = SPD data ready 0 = SPD data not available. See *SDRAM SPD memory* on page 4-30.
4:2	MEMSIZE	Read/write	These bits specify the size of the SDRAM module fitted to the core module. The bits are encoded as follows: 000 = 16MB 001 = 32MB 010 = 64MB (default) 011 = 128MB 100 = 256MB 101 = Reserved 110 = Reserved 111 = Reserved.
1:0	CASLAT	Read/write	These bits specify the CAS latency for the core module SDRAM. The bits are encoded as follows: 00 = Reserved 01 = Reserved 10 = 2 cycles (default) 11 = 3 cycles.

Copyright © 2000 ARM Limited. All rights reserved.

4.6.10 Core module initialization register

The core module initialization register (CM_INIT) is used to control the configuration pins of the ARM9x6E-S testchip (see *Test chip configuration control* on page 3-2) and the mapping of the on-board SRAM (see *Using REMAP* on page 4-7).



Note

Before writing to the CM_INIT register, you must unlock it by writing the value `0x0000A05F` to the CM_LOCK register. After writing the CM_INIT register, relock it by writing any value other than `0x0000A05F` to the CM_LOCK register.

Table 4-11 describes the core module initialization register bits. The ARM9x6E-S test chip samples the configuration values during reset. To reconfigure the test chip, set the required bit values shown in Table 4-11, and then press the reset button. The default settings are always in effect following a power-on reset.

Table 4-11 CM_INIT register

Bits	Name	Access	Function
31:24	USERIN[7:0]	Read	This bits connect with the USERIN[7:0] pins of the test chip.
23:18	Reserved	Use read-modify-write to preserve value.	
17	INITRAM (ARM966E-S only)	Read/write	This bit is used to enable or disable the TCRAM: 0 = TCRAM disabled (default after POR) 1 = TCRAM enabled. See *About the ARM9x6E-S memory map* on page 4-2.

Copyright © 2000 ARM Limited. All rights reserved.

Table 4-11 CM_INIT register (continued)

Bits	Name	Access	Function
16	SRAMMODE	Read/write	This bit controls the mapping of the on-board SSRAM: 0 = SSRAM mode 0 (default after POR) 1 = SSRAM mode 1. Emulates the TCRAM. See *SSRAM mode selection* on page 4-6
15:14	Reserved	Use read-modify-write to preserve value.	
13:8	CLKRATIO	Read/write	Used to set the internal clock multiplier. The default value after POR is `b000001` (multiply by 2). See *Test chip clocks* on page 3-22. The CLKRATIO multiplier only has any effect if PLLBYPASS bit is 0.
7	Reserved	Use read-modify-write to preserve value.	
6:4	HCLKDIV	Read/write	These bits are used to set the internal **HCLK** divider, to divide **CLK** by n+1. The default value after POR is `b000010` (divide by 3) giving a default local memory bus clock speed of 40MHz. See *Test chip clocks* on page 3-22.
3	Reserved	Use read-modify-write to preserve value.	
2	VINITHI	Read	This bit controls the mapping of the vectors 0 = vectors at 0 (default after POR) 1 = vectors at `0xFFFF0000`. See *Test chip configuration control* on page 3-2.
1	PLLBYPASS	Read	Current state on the PLLBYPASS pin of the testchip after a reset
0	PLLBYPASS	Write	Enables the PLL bypass feature of the testchip. The setting only takes effect after a reset. 0 = bypass OFF 1 = bypass ON (default after POR). See *Test chip configuration control* on page 3-2

Copyright © 2000 ARM Limited. All rights reserved.

4.6.11 Core module reference clock cycle counter

This register (CM_REFCNT) provides a 32-bit count value. The count increments at the fixed reference clock frequency of 24MHz and can be used as a real-time counter. The register is reset to zero by a reset.

 Copyright © 2000 ARM Limited. All rights reserved.

4.7 Core module flag registers

The core module flag registers provide you with two 32-bit register locations containing general purpose flags. You can assign any meaning to the flags. The flag registers are listed in Table 4-13.

Table 4-12 Core module flag registers

Register Name	Address	Access	Reset by	Description
CM_FLAGS	0x10000030	R	Reset	Core module flag register
CM_FLAGSS	0x10000030	W	Reset	Core module flag set register
CM_FLAGSC	0x10000034	W	Reset	Core module flag clear register
CM_NVFLAGS	0x10000038	R	POR	Core module non-volatile flag register
CM_NVFLAGSS	0x10000038	W	POR	Core module non-volatile flag set register
CM_NVFLAGSC	0x1000003C	W	POR	Core module non-volatile flag clear register

The core module provides two distinct types of flag registers:

- the flag register is cleared by a normal reset, such as a reset caused by pressing the reset button
- non-volatile flag register retain its contains after a normal reset and is only cleared by a *Power-On Reset* (POR).

4.7.1 Flag/non-volatile flag register

The status register contains the current state of the flags.

4.7.2 Flag/non-volatile flag set register

The flag set locations are used to set bits in the flag registers as follows:

- write 1 to SET the associated flag.
- write 0 to leave the associated flag unchanged.

4.7.3 Flag/non-volatile flag clear register

The clear locations are used to clear bits in the flag registers as follows:

- write 1 to CLEAR the associated flag
- write 0 to leave the associated flag unchanged

Copyright © 2000 ARM Limited. All rights reserved.

4.8 Core module interrupt registers

The core module provides a 3-bit IRQ controller and 3-bit FIQ controller to support the debug communications channel used for passing information between applications software and the debugger. The interrupt control registers are listed in Table 4-13.

Table 4-13 Interrupt controller registers

Register Name	Address	Access	Description
CM_IRQ_STAT	0x10000040	Read	Core module IRQ status register
CM_IRQ_RSTAT	0x10000044	Read	Core module IRQ raw status register
CM_IRQ_ENSET	0x10000048	Read/write	Core module IRQ enable set register
CM_IRQ_ENCLR	0x1000004C	Write	Core module IRQ enable clear register
CM_SOFT_INTSET	0x10000050	Read/write	Core module software interrupt set
CM_SOFT_INTCLR	0x10000054	Write	Core module software interrupt clear
CM_FIQ_STAT	0x10000060	Read	Core module FIQ status register
CM_FIQ_RSTAT	0x10000064	Read	Core module FIQ raw status register
CM_FIQ_ENSET	0x10000068	Read/write	Core module FIQ enable set register
CM_FIQ_ENCLR	0x1000006C	Write	Core module FIR enable clear register

Note

All registers are 32-bits wide and only support word-wide writes. Bits marked as *reserved* in the following sections should be preserved using read-modify-write operations.

The IRQ and FIQ controllers each provide three registers for controlling and handling interrupts. These are:

- status register
- raw status register
- enable register, which is accessed using the enable set and enable clear locations.

The way that the interrupt enable, clear, and status bits function for each interrupt is illustrated in Figure 4-6 on page 4-27 and described in the following subsections. The illustration shows the control for one IRQ bit. The remaining IRQ bits and FIQ bits are controlled in a similar way.

Copyright © 2000 ARM Limited. All rights reserved.

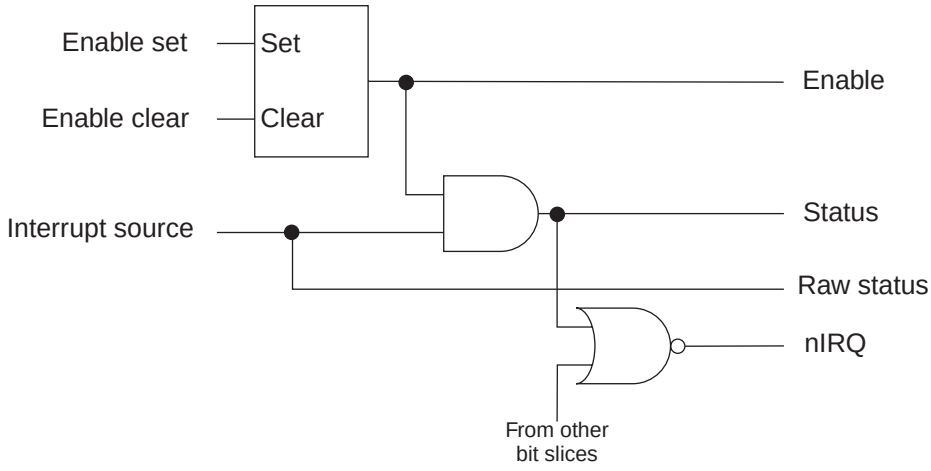


Figure 4-6 Interrupt control

4.8.1 IRQ/FIQ status register

The status register contains the logical AND of the bits in the raw status register and the enable register.

4.8.2 IRQ/FIQ raw status register

The raw status register indicates the signal levels on the interrupt request inputs. A bit set to 1 indicates that the corresponding interrupt request is active.

4.8.3 IRQ/FIQ enable set register

The enable set locations are used to set bits in the enable registers as follows:

- write 1 to SET the associated bit.
- write 0 to leave the associated bit unchanged.

Read the current state of the enable bits from the ENSET location.

4.8.4 IRQ/FIQ enable clear register

The clear set locations are used to set bits in the enable registers as follows:

- write 1 to CLEAR the associated bit
- write 0 to leave the associated bit unchanged

Copyright © 2000 ARM Limited. All rights reserved.

4.8.5 Interrupt register bit assignment

The bit assignments for the IRQ and FIQ status, raw status and enable register are shown in Table 4-14.

Table 4-14 IRQ and FIQ register bit assignment

Bit	Name	Function
31:3	Reserved	Write as 0. Reads undefined.
2	COMMTx	Debug communications transmit interrupt. This interrupt indicates that the communications channel is available for the processor to pass messages to the debugger.
1	COMMRx	Debug communications receive interrupt. This interrupt indicates to the processor that messages are available for the processor to read.
0	SOFT	Software interrupt

4.8.6 Soft interrupt set and soft interrupt clear registers

The core module interrupt controller provides a register for controlling and clearing software interrupts. This register is accessed using the software interrupt set and software interrupt clear locations. The set and clear locations are used as follows:

- Set the software interrupt by writing to the CM_SOFT_INTSET location:

  write a 1 to SET the software interrupt

  write a 0 to leave the software interrupt unchanged.

- Read the current state of the of the software interrupt register from the CM_SOFT_INTSET location. A bit set to 1 indicates that the corresponding interrupt request is active.

- Clear the software interrupt by writing to the CM_SOFT_INTCLR location:

  write a 1 to CLEAR the software interrupt.

  write a 0 to leave the software interrupt unchanged.

 Copyright © 2000 ARM Limited. All rights reserved.

The bit assignment for the software interrupt register is shown in Table 4-15.

Table 4-15 IRQ register bit assignment

Bit	Name	Function
31:1	Reserved	Write as 0. Reads undefined.
0	SOFT	Software interrupt

Note

The *software interrupt* described in this section is used by software to generate IRQs or FIQs. It should not be confused with the ARM SWI software interrupt instruction. See the *ARM Architecture Reference Manual*.

Copyright © 2000 ARM Limited. All rights reserved.

4.9 SDRAM SPD memory

This area of memory contains a copy of the SPD data from the SPD EEPROM on the DIMM. Because accesses to the EEPROM are very slow, the data is copied to this memory during board initialization to allow faster random access to the SPD data (see *Serial presence detect* on page 3-8). The SPD memory contains 256 bytes of data, the most important of which are as shown in Table 4-16.

Table 4-16 SPD memory contents

Byte	Contents
2	Memory type
3	Number of row addresses
4	Number of column addresses
5	Number of banks
31	Module bank density (MB divided by 4)
18	CAS latencies supported
63	Checksum
64:71	Manufacturer
73:90	Module part number

Check for valid SPD data as follows:

1. Add together all bytes 0 to 62.
2. Logically AND the result with `0xFF`.
3. Compare the result with byte 63.

If the two values match, then the SPD data is valid.

Note

A number of SDRAM DIMMs do not comply with the JEDEC standard and do not implement the checksum byte. The Integrator is not guaranteed to operate with non-compliant DIMMs.

The code segment shown in Example 4-1 on page 4-31 can be used to correctly setup and remap the SDRAM.

Copyright © 2000 ARM Limited. All rights reserved.

Example 4-1

```
CM_BASE     EQU    0x10000000     ; base address of Core Module registers
SPD_BASE    EQU    0x10000100     ; base address of SPD information

lightled
           ; turn on header LED and remap memory
           LDR    r0, =CM_BASE    ; load register base address
           MOV    r1,#5           ; set remap and led bits
           STR    r1,[r0,#0xc]    ; write the register
           ; setup SDRAM

readspdbit
           ; check SPD bit is set
           LDR    r1,[r0,#0x20]   ; read the status register
           AND    r1,r1,#0x20     ; mask SPD bit (5)
           CMP    r1,#0x20        ; test if set
           BNE    readspdbit      ; branch until the SPD memory has been read

setupsdram
           ; work out the SDRAM size
           LDR    r0, =SPD_BASE  ; point at SPD memory
           LDRB   r1,[r0,#3]     ; number of row address lines
           LDRB   r2,[r0,#4]     ; number of column address lines
           LDRB   r3,[r0,#5]     ; number of banks
           LDRB   r4,[r0,#31]    ; module bank density
           MUL    r5,r4,r3       ; calculate size of SDRAM (MB divided by 4)
           MOV    r5,r5,ASL#2    ; size in MB
           CMP    r5,#0x10       ; is it 16MB?
           BNE    not16          ; if no, move on
           MOV    r6,#0x2        ; store size and CAS latency of 2
           B      writesize

not16
           CMP    r5,#0x20       ; is it 32MB?
           BNE    not32          ; if no, move on
           MOV    r6,#0x6        ; store size and CAS latency of 2
           B      writesize

not32
           CMP    r5,#0x40       ; is it 64MB?
           BNE    not64          ; if no, move on
           MOV    r6,#0xa        ; store size and CAS latency of 2
           B      writesize
```


Copyright © 2000 ARM Limited. All rights reserved.

```
not64
            CMP     r5,#0x80    ; is it 128MB?
            BNE     not128      ; if no, move on
            MOV     r6,#0xe     ; store size and CAS latency of 2
            B       writesize

not128
            ; if it is none of these sizes then it is either 256MB, or
            ; there is no SDRAM fitted so default to 256MB.
            MOV     r6,#0x12    ; store size and CAS latency of 2

writesize
            MOV      r1,r1,ASL#8       ; get row address lines for SDRAM register
            ORR      r2,r1,r2,ASL#12   ; OR in column address lines
            ORR      r3,r2,r3,ASL#16   ; OR in number of banks
            ORR      r6,r6,r3          ; OR in size and CAS latency
            LDR      r0, =CM_BASE      ; point at module registers
            STR      r6,[r0,#0x20]     ; store SDRAM parameters
```

 Copyright © 2000 ARM Limited. All rights reserved.

Appendix A
Signal Descriptions

This index provides a summary of signals present on the core module main connectors. It contains the following sections:

- *HDRA* on page A-2
- *HDRB* on page A-4
- *Trace connector pinout* on page A-9
- *Logic analyzer connectors* on page A-10.

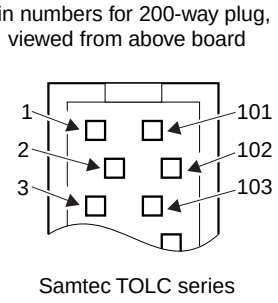
Note

For the Multi-ICE connector pinout and signal descriptions see *JTAG signals* on page 3-28.

Copyright © 2000 ARM Limited. All rights reserved.

A.1 HDRA

Figure A-1 shows the pin numbers of the HDRA plug and socket. All pins on the HDRA socket are connected to the corresponding pins on the HDRA plug.



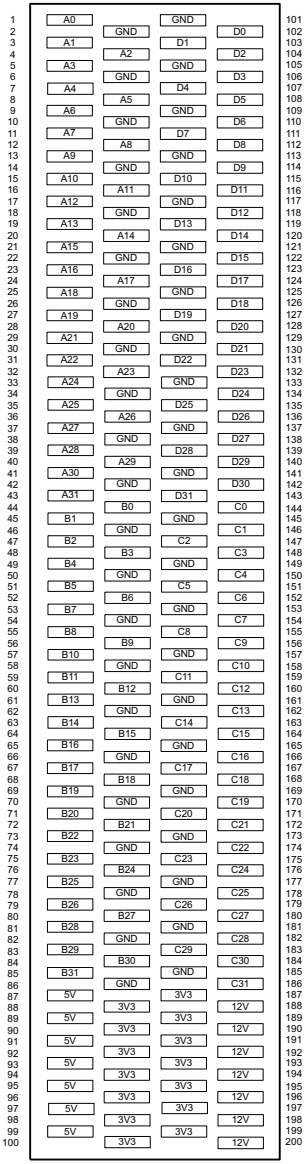

A0 GND GND D0
A1 A2 D1 D2
A3 GND GND D3
A4 A5 D4 D5
A6 GND GND D6
A7 A8 D7 D8
A9 GND GND D9
A10 A11 D10 D11
A12 GND GND D12
A13 A14 D13 D14
A15 GND GND D15
A16 A17 D16 D17
A18 GND GND D18
A19 A20 D19 D20
A21 GND GND D21
A22 A23 D22 D23
A24 GND GND D24
A25 A26 D25 D26
A27 GND GND D27
A28 A29 D28 D29
A30 GND GND D30
A31 B0 D31 C0
B1 GND GND C1
B2 B3 C2 C3
B4 GND GND C4
B5 B6 C5 C6
B7 GND GND C7
B8 B9 C8 C9
B10 GND GND C10
B11 B12 C11 C12
B13 GND GND C13
B14 B15 C14 C15
B16 GND GND C16
B17 B18 C17 C18
B19 GND GND C19
B20 B21 C20 C21
B22 GND GND C22
B23 B24 C23 C24
B25 GND GND C25
B26 B27 C26 C27
B28 GND GND C28
B29 B30 C29 C30
B31 GND GND C31
5V 3V3 3V3 12V
5V 3V3 3V3 12V
5V 3V3 3V3 12V
5V 3V3 3V3 12V
5V 3V3 3V3 12V
5V 3V3 3V3 12V
5V 3V3 3V3 12V
5V 3V3 3V3 12V

Figure A-1 HDRA plug pin numbering

Copyright © 2000 ARM Limited. All rights reserved.

The signals present on the pins labeled A[31:0], B[31:0], and C[31:0] are described in in Table A-1 for an AHB system bus.

Table A-1 Bus bit assignment (for an AMBA AHB bus)

Pin label	Signal	Description
A[31:0]	**HADDR[3:0]**	System address bus
B[31:0]	Not used	-
C[31:0]	**CONT[31:0]**	See below
C[31:16]	Not used	-
C15	**HLOCKALL**	Locked transaction
C[14:13]	**HRESP[1:0]**	Slave response
C12	**HREADY**	Slave ready
C11	**HWRITE**	Write transaction
C[10:8]	**HPROT[2:0]**	Transaction protection type
C[7:5]	**HBURST[2:0]**	Transaction burst size
C4	**HPROT[3]**	Transaction protection type
C[3:2]	**HSIZE[1:0]**	Transaction width
C[1:0]	**HTRAN[1:0]**	Transaction type
D[31:0]	**HDATA[31:0]**	System data bus

Copyright © 2000 ARM Limited. All rights reserved.

A.2 HDRB

The HDRB plug and socket have slightly different pinouts, as described below.

A.2.1 HDRB socket pinout

Figure A-2 shows the pin numbers of the socket HDRB on the underside of the core module, viewed from above the core module.

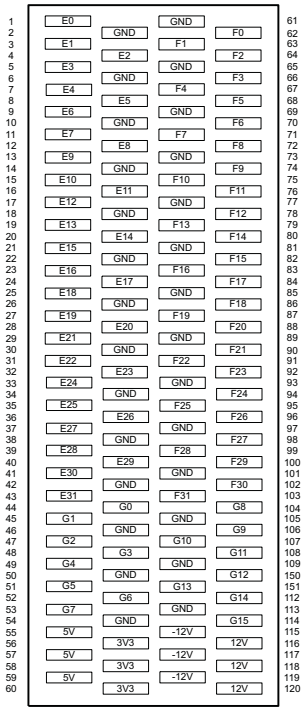


Figure A-2 HDRB socket pin numbering

Copyright © 2000 ARM Limited. All rights reserved.

A.2.2 HDRB plug pinout

Figure A-3 shows the pin numbers of the HDRB plug on the top of the core module.

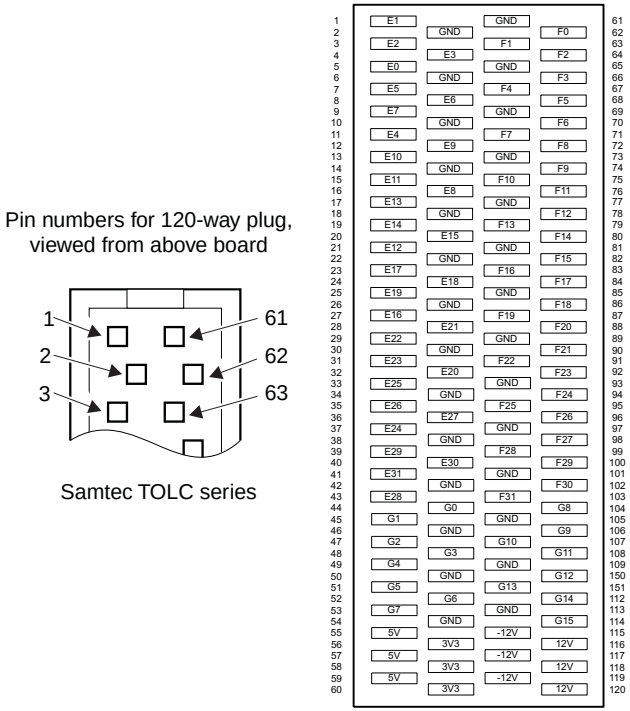


Figure A-3 HDRB plug pin numbering

Copyright © 2000 ARM Limited. All rights reserved.

A.2.3 Through-board signal connections

The signals on the pins labeled E[31:0] are cross-connected between the plug and socket so that the signals are rotated through the stack in groups of four. For example, the first block of four are connected as shown in Table A-2.

Table A-2 Signal cross-connections (example)

Plug		Socket
E0	connects to	E1
E1	connects to	E2
E2	connects to	E3
E3	connects to	E0

For details about the signal rotation scheme, see *System bus signal routing* on page 3-16.

The signals on the pins labeled F[31:0] are connected so that pins on the socket are connected to the corresponding pins on the plug.

The signals on G[16:8] and G[5:0] are connected so that pins on the socket are connected to the corresponding pins on the plug.

Pins G[7:6] carry the JTAG **TDI** and **TDO** signals. The signal **TDO** is routed through devices on each board as it passes up through the stack (see *JTAG signals* on page 3-28).

Copyright © 2000 ARM Limited. All rights reserved.

A.2.4 HDRB signal descriptions

Table A-3 describes the signals on the pins labeled E[31:0], F[31:0], and G[16:0] for AMBA AHB system bus.

Table A-3 HDRB signal description (AHB)

Pin label	Name	Description
E[31:28]	**SYSCLK[3:0]**	System clock to each core module/expansion card
E[27:24]	**nPPRES[3:0]**	Processor present
E[23:20]	**nIRQ[3:0]**	Interrupt request to processors 3, 2, 1, and 0 respectively
E[19:16]	**nFIQ[3:0]**	Fast interrupt requests to processors 3, 2, 1, and 0 respectively
E[15:12]	**ID[3:0]**	Core module stack position indicator
E[11:8]	**HLOCK[3:0]**	System bus lock from processor 3, 2, 1, and 0 respectively
E[7:4]	**HGRANT[3:0]**	System bus grant to processor 3, 2, 1, and 0 respectively
E[3:0]	**HBUSREQ[3:0]**	System bus request from processors 3, 2, 1, and 0 respectively
F[31:0]	-	Not connected
G16	**nRTCKEN**	RTCK AND gate enable
G[15:14]	**CFGSEL[1:0]**	FPGA configuration select
G13	**nCFGEN**	Sets motherboard into configuration mode
G12	**nSRST**	Multi-ICE reset (open collector)
G11	**FPGADONE**	Indicates when FPGA configurarion is complete (open collector)
G10	**RTCK**	Returned JTAG test clock
G9	**nSYSRST**	Buffered system reset
G8	**nTRST**	JTAG reset
G7	**TDO**	JTAG test data out

Copyright © 2000 ARM Limited. All rights reserved.

Table A-3 HDRB signal description (AHB) (continued)

Pin label	Name	Description
G6	**TDI**	JTAG test data in
G5	**TMS**	JTAG test mode select
G4	**TCK**	JTAG test clock
G[3:1]	**MASTER[2:0]**	Master ID. Binary encoding of the master currently performing a transfer on the bus. Corresponds to the module ID and to the **HBUSREQ** and **HGRANT** line numbers.
G0	**nMBDET**	Motherboard detect pin

Copyright © 2000 ARM Limited. All rights reserved.

A.3 Trace connector pinout

Table A-4 shows the pinout of the Trace connector. The ARM testchip ETMEXTOUT pins are routed to test points.

Table A-4 Trace connector pinout

Channel	Pin	Pin	Channel
no connect	1	2	no connect
no connect	3	4	no connect
GND	5	6	**TRACECLK**
DBGRQ	7	8	**DBGACK**
nSRST	9	10	**EXTTRIG**
TDO	11	12	**VDD** (3.3V)
RTCK	13	14	**VDD** (3.3V)
TCK	15	16	**TRACEPKT7**
TMS	17	18	**TRACEPKT6**
TDI	19	20	**TRACEPKT5**
nTRST	21	22	**TRACEPKT4**
TRACEPKT15	23	24	**TRACEPKT3**
TRACEPKT14	25	26	**TRACEPKT2**
TRACEPKT13	27	28	**TRACEPKT1**
TRACEPKT12	29	30	**TRACEPKT0**
TRACEPKT11	31	32	**TRACESYNC**
TRACEPKT10	33	34	**PIPESTAT2**
TRACEPKT9	35	36	**PIPESTAT1**
TRACEPKT8	37	38	**PIPESTAT0**

The ARM testchip **ETMEXTIN** pins are connected to the FPGA so that they could be driven by a special ETM validation FPGA configuration. In normal configurations the **ETMEXTTRIG** signal from the Trace Port connector is routed through the FPGA to the **ETMEXTIN[1]** pin.

Copyright © 2000 ARM Limited. All rights reserved.

A.4 Logic analyzer connectors

A logic analyzer connects to the local memory bus on the core module using high-density AMP Mictor connectors. There are four logic analyzer connectors labelled:

- *HADDR connector* on page A-11
- *CONTROL connector* on page A-12
- *HRDATA connector* on page A-13
- *HWDATA connector* on page A-14.

Note

HRDATA follows **HWDATA** on write cycles. If logic analyzer connections are limited HRDATA should be used.

The logic analyzer connection is a high-density AMP Mictor connector. The connectors carries 32 signals and 2 clocks or qualifiers. Figure A-4 shows the connector and identification of pin 1.

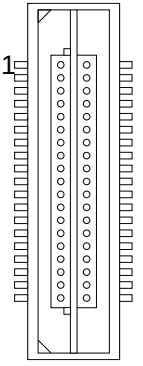


Figure A-4 AMP Mictor connector

Note

Agilent (formerly HP) and Tektronix label these connectors differently, but the assignments of signals to physical pins is appropriate for both systems and pin 1 is always in the same place. The schematic is labelled according to the Agilent pin assignment.

Copyright © 2000 ARM Limited. All rights reserved.

A.4.1 HADDR connector

Table A-5 shows the pinout of J9.

Table A-5 HADDR (J9)

Channel	Pin	Pin	Channel
no connect	1	2	no connect
GND	3	4	no connect
HTRANS1	5	6	**HCLK**
HADDR31	7	8	**HADDR15**
HADDR30	9	10	**HADDR14**
HADDR29	11	12	**HADDR13**
HADDR28	13	14	**HADDR12**
HADDR27	15	16	**HADDR11**
HADDR26	17	18	**HADDR10**
HADDR25	19	20	**HADDR9**
HADDR24	21	22	**HADDR8**
HADDR23	23	24	**HADDR7**
HADDR22	25	26	**HADDR6**
HADDR21	27	28	**HADDR5**
HADDR20	29	30	**HADDR4**
HADDR19	31	32	**HADDR3**
HADDR18	33	34	**HADDR2**
HADDR17	35	36	**HADDR1**
HADDR16	37	38	**HADDR0**

Copyright © 2000 ARM Limited. All rights reserved.

A.4.2 CONTROL connector

Table A-6 shows the pinout of J12.

Table A-6 Control (J12)

Channel	Pin	Pin	Channel
no connect	1	2	no connect
GND	3	4	no connect
nRESET	5	6	unused
nIRQ	7	8	unused
nFIQ	9	10	unused
HBUSREQ	11	12	unused
HGRANT	13	14	unused
HLOCK	15	16	unused
HPROT3	17	18	unused
HPROT2	19	20	unused
HRESP1	21	22	unused
HRESP0	23	24	unused
HREADY	25	26	unused
unused	27	28	unused
HSIZE1	29	30	**HBURST2**
HSIZE0	31	32	**HBURST1**
HPROT1	33	34	**HBURST0**
HPROT0	35	36	**HTRANS1**
HWRITE	37	38	**HTRANS0**

Copyright © 2000 ARM Limited. All rights reserved.

A.4.3 HRDATA connector

Table A-7 shows the pinout of J11.

Table A-7 HRDATA (J11)

Channel	Pin	Pin	Channel
no connect	1	2	no connect
GND	3	4	no connect
HREADY	5	6	**HCLK**
HRDATA31	7	8	**HRDATA15**
HRDATA30	9	10	**HRDATA14**
HRDATA29	11	12	**HRDATA13**
HRDATA28	13	14	**HRDATA12**
HRDATA27	15	16	**HRDATA11**
HRDATA26	17	18	**HRDATA10**
HRDATA25	19	20	**HRDATA9**
HRDATA24	21	22	**HRDATA8**
HRDATA23	23	24	**HRDATA7**
HRDATA22	25	26	**HRDATA6**
HRDATA21	27	28	**HRDATA5**
HRDATA20	29	30	**HRDATA4**
HRDATA19	31	32	**HRDATA3**
HRDATA18	33	34	**HRDATA2**
HRDATA17	35	36	**HRDATA1**
HRDATA16	37	38	**HRDATA0**

Copyright © 2000 ARM Limited. All rights reserved.

A.4.4 HWDATA connector

Table A-8 shows the pinout of J11.

Table A-8 HWDATA (J10)

Channel	Pin	Pin	Channel
no connect	1	2	no connect
GND	3	4	no connect
unused	5	6	**HCLK**
HWDATA31	7	8	**HWDATA15**
HWDATA30	9	10	**HWDATA14**
HWDATA29	11	12	**HWDATA13**
HWDATA28	13	14	**HWDATA12**
HWDATA27	15	16	**HWDATA11**
HWDATA26	17	18	**HWDATA10**
HWDATA25	19	20	**HWDATA9**
HWDATA24	21	22	**HWDATA8**
HWDATA23	23	24	**HWDATA7**
HWDATA22	25	26	**HWDATA6**
HWDATA21	27	28	**HWDATA5**
HWDATA20	29	30	**HWDATA4**
HWDATA19	31	32	**HWDATA3**
HWDATA18	33	34	**HWDATA2**
HWDATA17	35	36	**HWDATA1**
HWDATA16	37	38	**HWDATA0**

Copyright © 2000 ARM Limited. All rights reserved.

Appendix B
Specifications

This appendix contains the specifications for the ARM Integrator/CM9x6E-S core module. It contains the following sections:

- *Electrical specification* on page B-2
- *Timing specification* on page B-3
- *Mechanical details* on page B-6.

Copyright © 2000 ARM Limited. All rights reserved.

B.1 Electrical specification

This section provides details of the voltage and current characteristics for the core module.

B.1.1 Bus interface characteristics

Table B-1 shows the core module electrical characteristics for the system bus interface. The core module uses 3.3V and 5V sources. The 12V inputs are supplied by the motherboard but not used by the core module.

Table B-1 Core module electrical characteristics

Symbol	Description	Min	Max	Unit
3V3	Supply voltage (interface signals)	3.1	3.5	V
5V	Supply voltage	4.75	5.25	V
V_{IH}	High-level input voltage	2.0	3.6	V
V_{IL}	Low-level input voltage	0	0.8	V
V_{OH}	High-level output voltage	2.4	-	V
V_{OL}	Low-level output voltage	-	0.4	V
C_{IN}	Input capacitance	-	20	pF

B.1.2 Current requirements

Table B-2 shows the current requirments measured at room temperature and nominal voltage. These meaurements include the current drawn by Multi-ICE, which is approximately 160mA at 3.3V.

Table B-2 Current requirements

System	At 3.3V	At 5V
Standalone core module	1A	100mA
Motherboard (AP or SP) and one core module	1.5A	500mA

An Integrator/AP or SP with additional core or logic modules draws more current, and future core modules may require more current. For these reasons, provision is made to power the system with an ATX-type power supply.

Copyright © 2000 ARM Limited. All rights reserved.

B.2 Timing specification

The CM9x6E-S uses an AHB system bus. The timing parameters for this bus are shown in:

- Table B-3 *Clock and reset parameters*
- Table B-4 *AHB slave input parameters* on page B-4
- Table B-5 *AHB slave output parameters* on page B-4
- Table B-6 *Bus master input timing parameters* on page B-4
- Table B-7 *Bus master output timing parameters* on page B-5.

In addition notes on how the parameters are derived are given in *Notes on FPGA timing analysis* on page B-5.

B.2.1 Core module timing and the AMBA Specification

The parameters listed are those specified in the AMBA Specification with the following important differences:

- only output valid and input setup times are quoted
- the required input hold time (Tih) is always less than or equal to 0ns
- the output hold time (Toh) is always greater than 2ns.

Each version and revision of the FPGA has subtly different timing. The typical figures shown in Table B-3 are those you can expect under nominal conditions and should be used as a guideline when designing your own motherboards and modules. The typical figures have been rounded to simplify timing analysis and constraints.

B.2.2 Timing parameter tables

Table B-3 shows the clock and reset timing parameters.

Table B-3 Clock and reset parameters

Parameter	Description	Typ
Tclk	**HCLK** clock period	30
Tisrst	**HRESETn** deasserted setup time before **HCLK**	15

Copyright © 2000 ARM Limited. All rights reserved.

Table B-4 shows the AHB slave input parameters.

Table B-4 AHB slave input parameters

Parameter	Description	Typ
Tistr	Transfer type setup time before **HCLK**	5
Tisa	**HADDR[31:0]** setup time before **HCLK**	10
Tisctl	**HWRITE**, **HSIZE[2:0]** and **HBURST[2:0]** setup time before HCLK	5
Tiswd	Write data setup time before **HCLK**	5
Tisrdy	Ready setup time before **HCLK**	5

Table B-5 shows the AHB slave output parameters.

Table B-5 AHB slave output parameters

Parameter	Description	Typ
Tovrsp	Response valid time after **HCLK**	15
Tovrd	Data valid time after **HCLK**	15
Tovrdy	Ready valid time after **HCLK**	15

Table B-6 shows the bus master input parameters.

Table B-6 Bus master input timing parameters

Parameter	Description	Typ
Tisgnt	**HGRANTx** setup time before **HCLK**	5
Tisrdy	Ready setup time before **HCLK**	5
Tisrsp	Response setup time before **HCLK**	5
Tisrd	Read data setup time before **HCLK**	5

Copyright © 2000 ARM Limited. All rights reserved.

Table B-7 shows the bus master outputtiming parameters.

Table B-7 Bus master output timing parameters

Parameter	Description	Typ
Tovtr	Transfer type valid time after **HCLK**	15
Tova	Address valid time after **HCLK**	15
Tovctl	**HWRITE**, **HSIZE[2:0]** and **HBURST[2:0]** valid time after HCLK	15
Tovwd	Write data valid time after **HCLK**	15
Tovreq	Request valid time after **HCLK**	15
Tovlck	Lock valid time after **HCLK**	15

B.2.3 Notes on FPGA timing analysis

The system bus on all Integrator boards is routed between FPGAs. These FPGAs are routed with timing constraints similar to those shown the tables in *Timing parameter tables* on page B-3. The exact performance of a system depends on the timing parameters of the motherboard and all modules in the system. Some allowance also needs to be made for clock skew, routing delays and number of modules (that is, loading).

Not all FPGAs meet the ideal timing parameters, due to the complexity of the design or routing congestion within the device. For this reason, the PLL clock generators on Integrator default to a safe low value that all modules can achieve.

A detailed timing analysis involves calculating the input/output delays between modules for all timing parameters. In general, the simplest approach to determine the maximum operating frequency is to increase the frequency of the clock generators until the system becomes unstable.

Copyright © 2000 ARM Limited. All rights reserved.

B.3 Mechanical details

The core module is designed to be stackable on a number of different motherboards. Its size allows it to be mounted onto a CompactPCI motherboard while allowing the motherboard to be installed in a card cage.

Figure B-1 shows the mechanical outline of the core module.

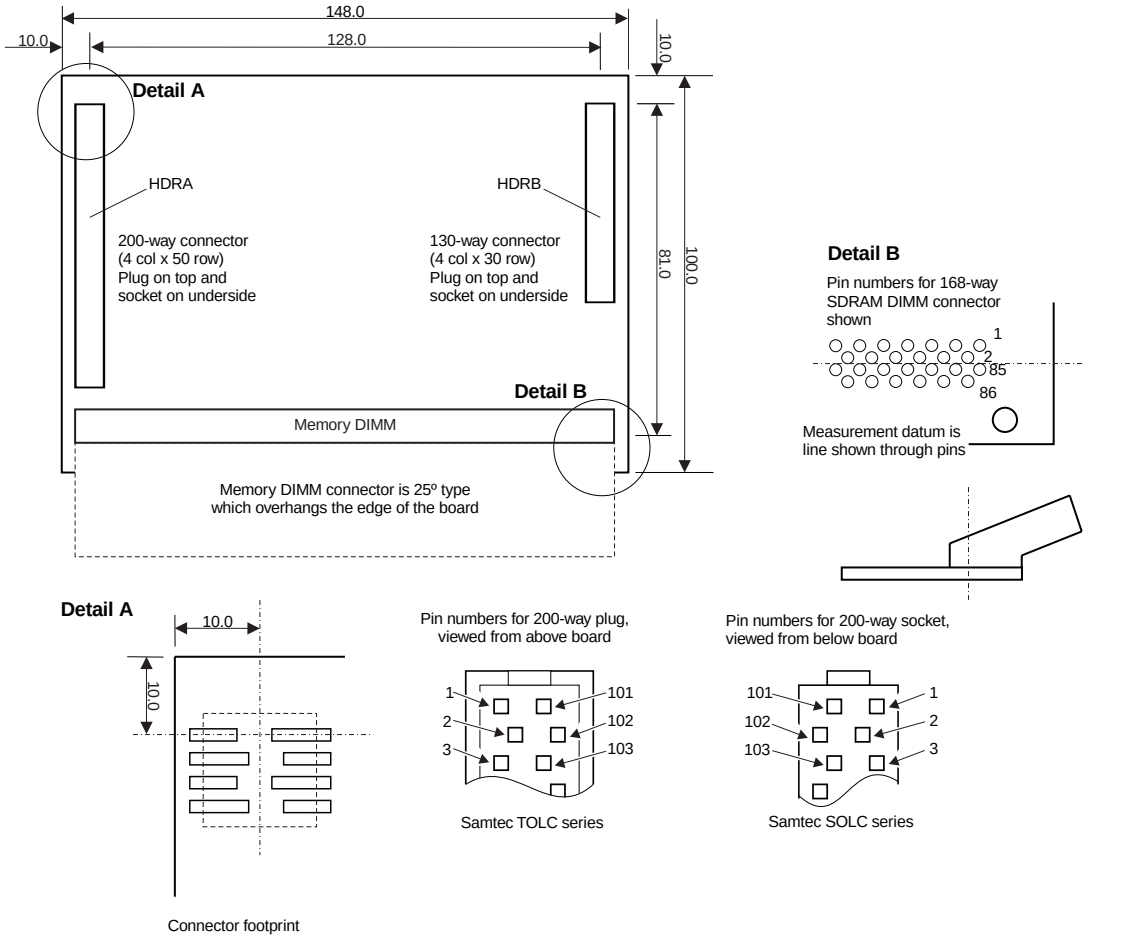


Figure B-1 Board outline

Copyright © 2000 ARM Limited. All rights reserved.

Index

The items in this index are listed in alphabetical order, with symbols and numerics appearing at the end. The references given are to page numbers.

A

About this book
  feedback xi
  typographical conventions ix
Access arbitration, SDRAM 3-8
Accesses
  boot ROM 4-7
  SDRAM 4-9
  SSRAM 4-7
Address decoding, module 3-18
Alias SDRAM addresses 4-10
ARCH bits 4-14
Architecture, system 1-5
ARM processor, overview 1-5
ARM publications, related x
Assembled Integrator system 2-5
AUX CLK test point 1-11
AUX oscillator register 4-18
Auxiliary clock 3-20
AUX_OD bits 4-19
AUX_RDW bits 4-19
AUX_VCW bits 4-19

B

bits 4-22
Block diagram 1-5
Board layout 1-3
Board outline B-6
Boot ROM, accesses 4-7
BUILD bits 4-14
Bus bridge, system 3-12
Bus mode bits 4-15

C

Calculate the clock frequencies 3-21
CAS latency, setting 4-21
CASLAT bits 4-21
CE Declaration of Conformity ii
CFGEN LED 1-9
Checking for valid SPD data 4-30
CLKRATIO bits 4-23
Clock generator 1-7, 3-20
Clocks
  auxiliary 3-20
  FPGA 3-20
  processor 3-20
Clocks, programming 3-21
CM_AUXOSC register 3-20, 4-18
CM_CTRL register 4-15
CM_FIQ_ENCLR register 4-26
CM_FIQ_ENSET register 4-26
CM_FIQ_RSTAT register 4-26
CM_FIQ_STAT register 4-26
CM_ID Register 4-13
CM_ID register 4-10
CM_INIT register 3-2, 3-20, 4-22
CM_IRQ_ENCLR register 4-26
CM_IRQ_ENSET register 4-26
CM_IRQ_RSTAT register 4-26
CM_IRQ_STAT register 4-26
CM_LMBUSCNT register 4-18
CM_LOCK register 4-14, 4-17, 4-22
CM_OSC register 3-20, 4-14
CM_PROC register 4-14
CM_REFCNT register 4-24
CM_SDRAM 3-9

Copyright © 2000 ARM Limited. All rights reserved.

CM_SDRAM register 4-20
CM_SOFT_INTCLR register 4-26
CM_SOFT_INTSET register 4-26
CM_SPD 3-8
CM_STAT register 4-16
CONFIG LED 1-8, 3-24
CONFIG link 1-8, 3-24
Configuration
  processor 1-5
Configuration mode 3-27
Configuring the memory map 4-4
Connecting
  Multi-ICE 2-4
Connecting power 2-3
Connector
  Logic analyzer A-10
Connector locations 1-3
Connectors
  HDRA 3-16
  HDRB 3-16
  Multi-ICE 2-4
  power 2-3
Controllers
  clock 1-7
  FIQ 4-26
  IRQ 4-26
  reset 1-6, 3-10
  SDRAM 3-8
  SSRAM 3-5
Core clock VCO divider 4-15
Core module control register 4-15
Core module FPGA 1-5
Core module ID 2-6
  ID selection 3-18
  ID signals 3-18
Core module local memory bus cycle counter 4-18
Core module registers 4-12
Core module, stack position 4-16

D

Damage, preventing 1-12
Debug comms channel 4-26
Debugging modes 3-27
Document confidentiality status iii
DONE LED 1-9

E

EFIBYPASS signal 3-4
Electrical characteristics B-2
Electromagnetic conformity ii
Enable register, flag 4-25
Enable register, interrupt 4-27
Ensuring safety 1-12

F

FCC notice ii
FIFOs 3-12
FIQ controller 4-26
Fitting SDRAM 2-2
Flag registers 4-25
FPGA 1-5
FPGA bits 4-14
FPGA clock 3-20
Function of the LEDs 1-9

G

Global SDRAM 4-10
Ground points 1-10

H

HCLK test point 1-11
HCLKDIV bits 4-23
HDRA connector 3-16
HDRA pinout A-2
HDRA/HDRB connectors 1-3
HDRB connector 3-16
HDRB plug pinout A-5
HDRB signals A-7
HDRB socket pinout A-4

I

ICS525 clock generators 3-21
ID bits, Core module ID, reading 4-17
Indicators, function 1-9
INITRAM bit 4-22
INITRAM signal 3-3
Interrupt control 4-27
Interrupt register bit assignment 4-28
Interrupt registers 4-26
Interrupt, signal routing 3-19
Interupt status 4-27
IRQ and FIQ register bit assignment 4-28
IRQ controller 4-26

J

JTAG 3-24
JTAG debug 1-7
JTAG scan path 3-25
JTAG signals 3-28
JTAG, connecting 2-4

L

Layout, board 1-3
LEDs, function 1-9
Links
  CONFIG 1-8
Local SDRAM 4-9
LOCKED bit 4-18
LOCKVAL bits 4-18
Logic analyzer connector A-10
Logic analyzer connectors 1-3

M

MAN bits 4-14
MBDET bit 4-16
Memory map, configuring 4-4
MEMSIZE 4-21
MICEBYPASS signal 3-4
Microprocessor core 3-2
MISC LED 1-9
MISC LED control 4-16
Module ID selection 3-18
Motherboard, attaching the core module 2-5
Multi-ICE 1-7, 3-24
  connecting 2-4
Multi-ICE connector 1-3

Copyright © 2000 ARM Limited. All rights reserved.

N

NBANKS bits 4-20
NCOLS bits 4-20
nEPRES signals 3-18
nMBDET signal 3-26
Normal debug mode 3-27
Notices, FCC ii
nPRES signals 3-18
NROWS bits 4-20

O

Oscillator register 4-14
Output divider 3-21
Overview, system 1-4

P

Pinout
  HDRA A-2
  HDRB plug A-5
  HDRB socket A-4
  Trace A-9
PLL LOCK test point 1-11
PLLBYPASS bit 3-22, 4-23
PLLBYPASS signal 3-3
PLLCKIN test point 1-11
PLLCLKIN signal 3-3
POWER 1-9
Power connector 2-3
Powering an attached core module 2-6
Precautions 1-12
Preventing damage 1-12
Processor clock 3-20
Processor configuration 1-5, 3-2
Processor reads 3-13
Processor register 4-14
Processor writes 3-12
Product feedback xi
Product status iii

R

Raw status register, interrupt 4-27
Reads by the processor from the system bus 3-13
REF CLK test point 1-11
REFCLK 3-20
Register addresses 4-12
Registers 1-6
  CM_AUXOSC 3-20, 4-18
  CM_CTRL 4-15
  CM_FIQ_ENCLR 4-26
  CM_FIQ_ENSET 4-26
  CM_FIQ_RSTAT 4-26
  CM_FIQ_STAT 4-26
  CM_ID 4-10, 4-13
  CM_INIT 3-2, 3-20, 4-22
  CM_IRQ_ENCLR 4-26
  CM_IRQ_ENSET 4-26
  CM_IRQ_RSTAT 4-26
  CM_IRQ_STAT 4-26
  CM_LMBUSCNT 4-18
  CM_LOCK 4-17
  CM_OSC 3-20, 4-14
  CM_PROC 4-14
  CM_REFCNT 4-24
  CM_SDRAM 4-20
  CM_SOFT_INTCLR 4-26
  CM_SOFT_INTSET 4-26
  CM_STAT 4-16
Related publications x
REMAP bit 4-16
REMAP operation 4-7
Remap, effect of 4-7
REMAP, operation 4-7
RESBYPASS signal 3-4
RESET bit 4-16
Reset controller 1-6, 3-10

S

Safety 1-12
SDRAM 3-14
  accesses 4-9
  controller 1-6, 3-8
  fitting 2-2
  global access 4-10
  operating mode 3-8
  operating without 2-2
  SPD memory 3-8, 4-30
SDRAM HCLK test point 1-11
SDRAM status and control register 4-20
Serial presence detect 3-8
Setting SDRAM size 4-21
Setup
  power connections 2-3
  standalone 2-2
Signal
  PLLCLKIN 3-3
Signal routing, interrupts 3-19
Signals
  EFIBYPASS 3-4
  INITRAM 3-3
  MICEBYPASS 3-4
  PLLBYPASS 3-3
  RESBYPASS 3-4
  VINITHI 3-3
Signals nEPRES 3-18
signals nPRES 3-18
SI_ID bits 4-17
Software interrupt registers 4-28
Software reset 4-16
SPD memory 3-8, 4-30
SPDOK bit 4-21
SRAMMODE bit 4-23
SSRAM alias 4-8
SSRAM controller 3-5
SSRAM modes 4-6
SSRAMSIZE bits 4-17
Standalone core module 2-2
Status and configuration registers 1-6
Status register 4-16
Status register, flag 4-25
Status register, interrupt 4-27
Supplying power 2-3
System architecture 1-5
System bus bridge 1-6, 3-12
System bus reads of the SDRAM 3-15
System bus signal routing 3-16
System bus writes to SDRAM 3-14
System overview 1-4

T

TDI signal 3-25
Test chip clocks 3-22
Test chip, overview 1-5
Test points 1-10

Copyright © 2000 ARM Limited. All rights reserved.

Through-board signals A-6
Tightly-Coupled RAM 4-2
Timing specification B-3
Trace connector 1-3
Trace connector pinout A-9
Typographical conventions ix

U

Using the core module with a
motherboard 2-5

V

VCO divider 3-21
VINITHI bit 4-23
VINITHI signal 3-3
Volatile memory 1-7

W

Writes by the processor to the system
bus 3-12

Numerics

Signals
ID 3-18
USERIN 4-22

 Copyright © 2000 ARM Limited. All rights reserved.